平成28年度修士論文

月面閃光観測用スペクトルカメラ

電気通信大学大学院　情報理工学研究科
情報・通信工学専攻　電子情報システムコース

学籍番号: 1531021
氏名: 柿沼 文広

指導教員:　柳澤 正久　教授

提出日　平成29年1月30日(月)

概要

月面閃光とは、月面で流星体が高速度で衝突した際に閃光が生じる現象のことである。月面閃光は地球近傍の小惑星のサイズ分布を知る手がかりになる。また、将来的に月震の震源位置や発生時刻の特定に役立つだろう。月面閃光の観測は他機関でも行われているが、閃光の色情報については未だ分かっていない。色情報が分かれば、その波長で感度の高い機材を使用することで月面閃光の観測が容易になる。従って、今回は月面閃光の色情報を得るため、月面閃光観測用にスペクトルカメラを作成した。そして、実際にスペクトルカメラを望遠鏡に取り付けて観測を行った。その結果、月面閃光のスペクトルであると思われる画像が 1 枚得られた。恒星と比較する事で閃光のスペクトルを求めた。

目 次

第1章　序論

1.1　研究背景・目的

宇宙空間には無数のメテオロイド (太陽系小天体) が飛び交っている。メテオロイドの衝突により地球では流星や火球と言った現象が発生するが、地球以外の天体においても衝突による発光現象が起きている。メテオロイドが月面に高速度で衝突した際には月面閃光と呼ばれる閃光が生じる。月面閃光は地球近傍におけるメテオロイドのサイズ分布調査において役立っている。また、将来的に月震 (月で起きる地震) の震源位置や発生時刻の特定という点で役立つだろう。そのため、他機関でも観測が行われており、NASA によって 2006 年から 2014 年の間に 300 個以上の月面閃光が観測された [1]。しかし、観測にはモノクロカメラが用いられているため、未だその色情報、スペクトルについては分かっていない。閃光のスペクトルが分かれば、その波長域に特化した観測機材を使用することで、今後の観測が容易になるだろう。また、閃光が黒体放射であれば、プランクの公式を用いて閃光の温度を推定する事も可能である。従って今回は月面閃光のスペクトルを調べる事を目的とし、スペクトルカメラを作成し、実際に観測を行った。

第2章　方法

2.1　スペクトルカメラの作成

モノクロ CMOS デジタルカメラ (ZWO 社 ASI174MM) に回折格子 (Edmund Optics 社 #46-067, 格子周波数 70 本/mm, ブレーズドタイプ) を直付けすることにより、スペクトルカメラを作成した。図 2.1 に作成したスペクトルカメラ及びその概略図を示す。このカメラは 12 ビットの階調で最大 1936×1216 の画素数で秒間 128 枚撮影する事が出来る。格子周波数 70 本/mm という値は、一般的に使われる回折格子の周波数よりも小さい。格子周波数が小さい回折格子の利点は、波長分散を小さくし、狭い範囲に光を集中させる事によって多くの光量を確保出来る点である。また、ブレーズド回折格子は特定の次数や波長で高い回折効率を示すという特徴を持っている。このタイプの回折格子は通常、格子が刻まれていない基板面から平行光を入射して使用する。しかし、基板面とカメラセンサーのカバーガラスを光学グリスを用いて接着させたため、格子側から光を入射した。回折格子がずれないように木材のバルサを削って作成した固定台をカメラ内に装着した。また、作った固定台は光の散乱を防ぐため、レンズ墨塗り用剤 (有限会社フィットマックロン) を用いて黒色に塗装した。一般的に、回折格子に入射させるのは平行光であるが、観測装置の簡単化のために、望遠鏡からの収束光を直接入射させた。

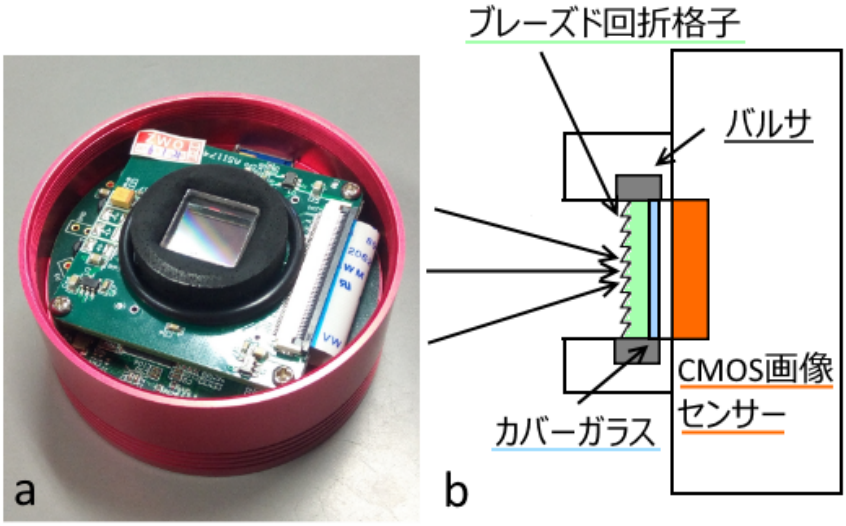

図 2.1: (a) 実際のスペクトルカメラ. 及び (b) スペクトルカメラの概略図. モノクロ CMOS デジタルカメラの撮像センサーのカバーガラスとブレーズド回折格子 (12.7 mm×12.7 mm、格子周波数 70 本/mm) の基板面を光学グリスで接着した. 回折格子がずれないよう、回折格子の周りに木材のバルサを削って作った固定台をはめ込んだ. また、作成した固定台はレンズ墨塗り用剤を用いて黒色に塗装した. このカメラでは 12 ビットの階調で最大 1936 × 1216 の画素で撮影が可能である.

2.2 波長キャリブレーション実験

スペクトルカメラの分光感度特性を求めるため、モノクロメーターを通した光をスペクトルカメラで撮影した。実験の実際の様子及び概略図を図 2.2 に示す。まず、ハロゲン光源 Ocean Optics 社 LS-1 を用いて、白色光を発生させた。その光を光ファイバーに通して、モノクロメーターに入射させ、再び別の光ファイバーに通し、全長 50 cm 程の人工星発生装置 (平行光を発生させる装置、本研究室 山本氏作成) に入射させた。そして、その光を焦点距離 8 mm のカメラ・レンズにより集光させ、作成したスペクトルカメラにて撮影した。モノクロメーターの波長を 50 nm 毎に 400 nm から 800 nm に設定し、撮影を行った。撮影時のカメラのゲインは 0 dB に固定、シャッタースピードは波長毎に調整しながら撮影を行った。撮影した画像の連続 10 フレームの画像を平均し、1 次像の写っている箇所の画素値を合計し、バックグラウンドの値を引いて 1 次像のカウントとした。バックグラウンドの値には一次像の周囲の数百画素の画素値を平均した値を用いた。

次にスペクトルカメラの代わりにフォトダイオードモジュール (浜松ホトニクス社 C10439-03) を取り付け、波長毎にモノクロメーターからの出力電圧を測定した。また、バックグラウンドの電圧として光源を点けない状態で電圧を測定した。測定

した出力電圧をフォトダイオードモジュールの分光感度特性 (mV/nW) で割ることによりモノクロメーターの放射束エネルギーを求めた。求めたカウントを 1 秒あたりに換算し、放射束エネルギーで割り、波長感度特性を求めた。求めたカウントと波長感度特性を表 2.2 に示す。また、波長感度特性のグラフを図 2.5 に示す。

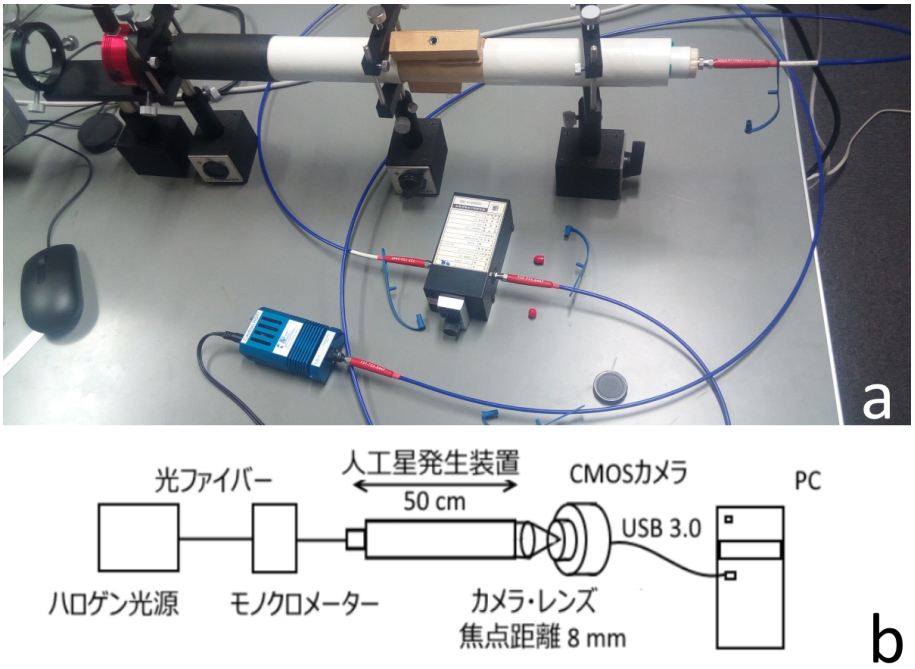

図 2.2: (a) 性能評価実験の様子. 及び (b) 概略図. Ocean Optics 社の LS-1 タングステンハロゲン光源 (a: 青色の装置) で発生させた白色光を光ファイバー (a: 青いケーブル) に通して Edmund Optics 社の 37598-K モノクロメーター (a: 中央の黒い装置) に入射し、再び別の光ファイバーに通して、人工星発生装置に入射させた. その光を焦点距離 8 mm のカメラ・レンズで集光させ、スペクトルカメラで撮影した. 尚、実際には部屋の明かりを消して実験を行った.

スペクトルカメラでモノクロメーターからの光を撮影した写真に、画像処理でそれぞれの波長に合った色をつけたものを図 2.3 に示す。図 2.3 は一番上から下へそれぞれ 50 nm 毎に 400 nm から 800 nm の波長に対応している。撮影した画像からそれぞれの波長の光が 0 次像から何画素目に写り込んでいるかを調べ、横軸を波長 (nm)、縦軸を 0 次像からの画素数 (個) として図 2.4 のグラフにまとめた。その結果、400 nm から 800 nm の波長域では 1 画素あたり 20 nm の光が写っていることが分かった。

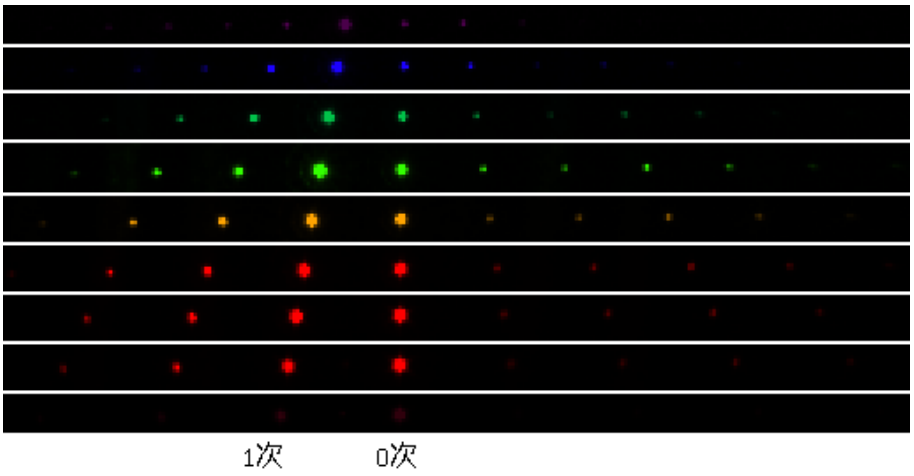

図 2.3: スペクトルカメラでモノクロメーターからの光を撮影した様子の擬似的なカラー写真．実際はモノクロ写真であるが、イメージしやすいように画像処理でそれぞれの波長の色で色付けした．一番上から下へそれぞれ、50 nm 毎に 400 nm から 800 nm の波長に対応している．

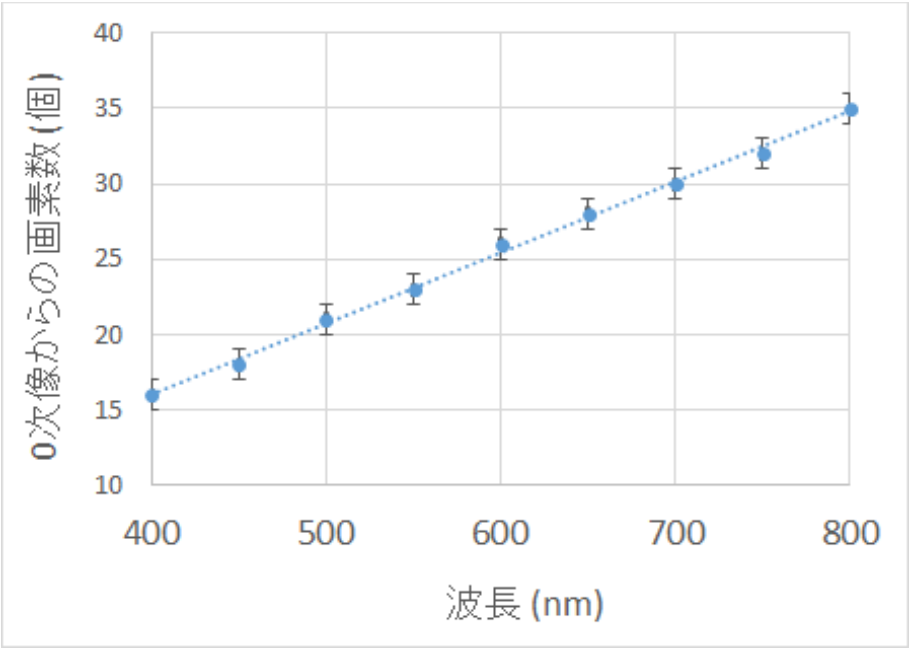

図 2.4: 光の波長と 0 次像からの画素数の関係．400 nm から 800 nm の波長域においては、1 画素あたり 20 nm 程の光が写っていることが分かった

表 2.1: 分光感度特性.

波長 (nm)	カウント (/s)	モノクロメーター出力 (nW)	カウント/放射束エネルギー (s^{-1} nW^{-1})
400	7.79.E+03	1.37.E-03	5.7.E+06
450	3.89.E+04	1.23.E-03	3.2.E+07
500	9.03.E+04	1.04.E-03	8.7.E+07
550	1.42.E+05	9.31.E-04	1.5.E+08
600	1.39.E+05	8.44.E-04	1.7.E+08
650	9.92.E+04	7.71.E-04	1.3.E+08
700	7.56.E+04	6.67.E-04	1.1.E+08
750	4.34.E+04	6.10.E-04	7.1.E+07
800	2.17.E+04	5.45.E-04	4.0.E+07

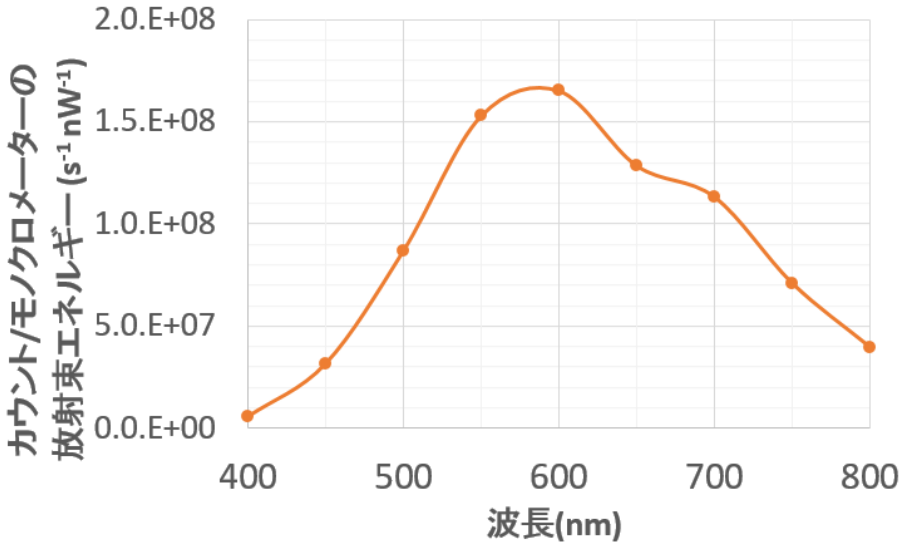

図 2.5: 分光感度特性. 1 秒あたりの 1 次像のカウントを求め、フォトダイオードモジュールで測定したモノクロメーターの放射束エネルギー (nW) で割った。

2.3 月面閃光の観測

観測場所は電気通信大学東 3 号館屋上の天体ドーム (世界測地系 北緯 35°39'28", 東経 139°32'37"(Google Map にて測定)) である。使用した望遠鏡は口径約 45 cm のニュートン式望遠鏡 (JMI Telescopes 社 NGT-18) である。望遠鏡にスペクトルカメラを装着し、USB3.0 ケーブルでワークステーション (Dell 社 Precision Tower

3620, CPU: Intel Xeon Processor E3-1240 v5 3.5 GHz, メモリ: 16 GB) と接続し、録画ソフト FireCapture Ver. 2.4 (Torsten Edelmann 氏提供のフリーウェア) を用いて、HDD に録画した。図 2.6 に観測装置の概略図を示す。カメラのゲインや露光時間の設定は観測の状況に合わせて変更し、ガンマは常にオフに設定した。観測した動画から閃光を発見するのには、自作の検出ソフトウェアを用いた。

月面閃光の観測は月面の夜側を観測するため、月の満ち欠けによって期間が限られる。観測は主に毎月、新月の次の日から上弦の月の日までの間 (月齢で言えば 2 から 8 程度) で晴天の夜に行った。時間帯は日が沈んで周囲が完全に暗くなってから、月の高度が約 15° に下がるまでである。雲が薄く掛かるような状態でも、カメラにはっきりと写り込んでしまい、誤検出の原因となるため晴天が望ましい。下弦の月の日から新月の前日まで観測する事も可能だが、時間帯が深夜から朝方になってしまうため、基本的には観測を行わなかった。実際に観測を行った日時を表 2.2 に示す。2017 年 1 月の観測はしぶんぎ座流星群による月面閃光を期待して観測を行ったものである。一方、その他の日程は散在流星群による月面閃光を期待して観測を行ったものである。

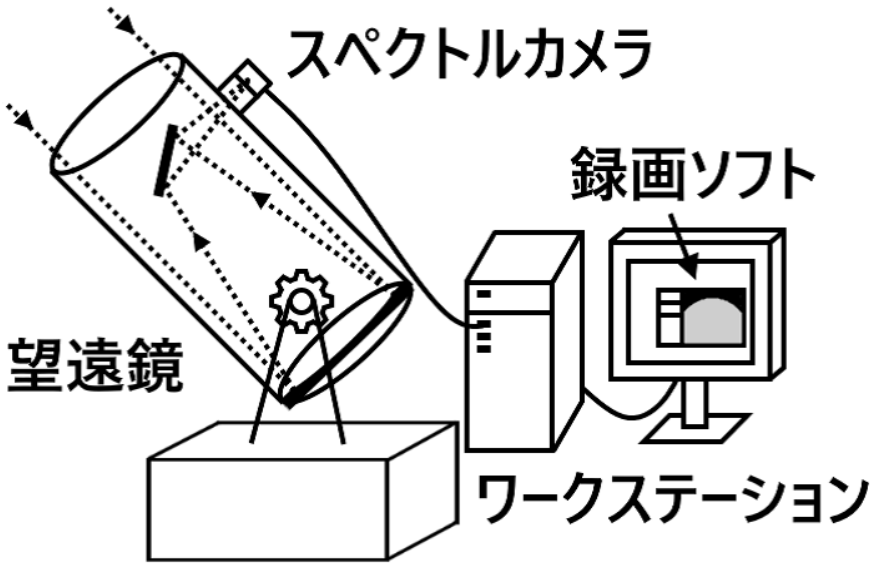

図 2.6: 観測装置概略図. 望遠鏡 (JMI Telescopes 社 NGT-18、口径 45 cm) にスペクトルカメラを取り付け、USB3.0 ケーブルでワークステーション (Dell 社 Precision Tower 3620, CPU: Intel Xeon Processor E3-1240 v5 3.5 GHz, メモリ: 16GB) と接続し、録画ソフト FireCapture Ver. 2.4 (Torsten Edelmann 氏提供) を用いて、HDD に録画した.

表 2.2: 観測実施日時 (JST).

日付	時間	月齢
2016 年 10 月 25 日	2:50 - 4:20	24
2016 年 11 月 03 日	17:30 - 18:00	4
2016 年 11 月 05 日	17:30 - 19:30	6
2016 年 11 月 07 日	17:50 - 20:30	8
2016 年 12 月 06 日	17:00 - 21:00	7
2016 年 12 月 07 日	20:45 - 22:15	8
2017 年 01 月 01 日	17:00 - 17:20	3
2017 年 01 月 03 日	17:20 - 20:20	5
2017 年 01 月 05 日	17:15 - 22:10	7

2.4 検出ソフトウェア

月面閃光の継続時間は衝突体のサイズによるが、1 秒にも満たない事が多い。そのため観測した動画から閃光のあるフレームをソフトウェアを用いて検出するのが好ましい。今回、2007 年に本研究室の池上氏 [2] が作成した閃光検出ソフトウェア newdetect を改良し、検出処理に用いた。このソフトウェアを以後、Serdetect と呼ぶ。実際の Serdetect の様子を図 2.7 に、ソフトウェアの動作をフローチャートで表したものを図 2.8 に示す。また、ソースコードを付録として、本書の最後に添付する。Serdetect は SER 形式の動画に対応した閃光検出ソフトウェアである。基本的な動作は次のようになっている。

(1) ファイルの読み込み

Serdetect はファイルを複数読み込み、検出処理をまとめて順に行う事が出来る。図 2.7 の左下に表示されたリストが簡易的エクスプローラーになっていて、そこから動画ファイルを読み込むことが出来る。また、読み込んだ際にもう一つウィンドウが表示され、動画を 1 フレームずつ閲覧出来るようになっている。

(2) 検出条件の設定

開始フレームや終了フレーム、検出上限フレーム数、処理開始時の閾値はファイル毎に設定が可能である。また、自動的に閾値を 0.5 下げて検出処理を繰り返したり、3 × 3 の画素でメディアンフィルタを掛けてから処理を行う事も可能である。宇宙線によりある一点がとても明るく映る事があり、誤検出の原因となる。メディアンフィルタはこのような宇宙線による誤検出を防ぐ事が出来る。

(3) 検出処理

検出ボタンを押すと検出処理が開始される。検出条件を満たすフレームがあれば、検出箇所の周辺に赤い四角を描画し、BMP 形式の画像で保存する。また、検出箇所の座標や画素値などの情報を含むテキストファイルが保存される。

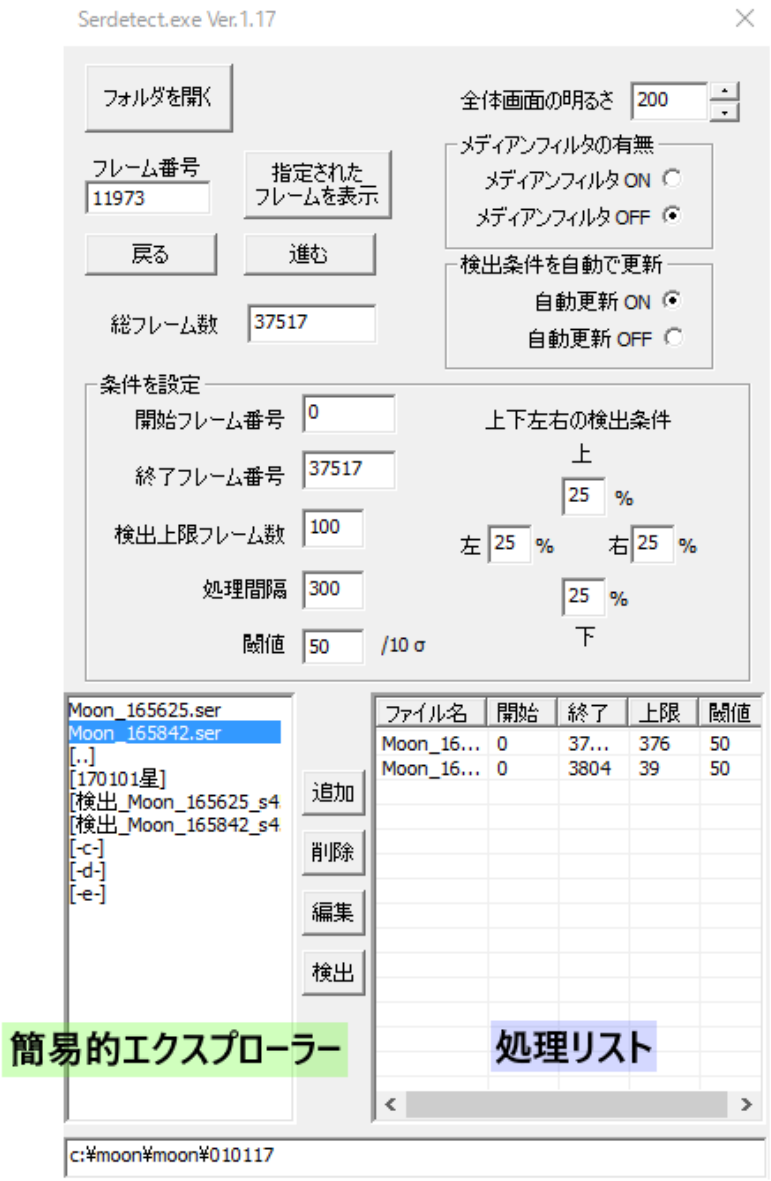

図 2.7: SER 形式動画用閃光検出ソフトウェア Serdetect. 簡易的エクスプローラーから SER 形式の動画ファイルを読み込む事が出来る. 開始フレームや終了フレーム、検出上限フレーム数、処理開始時の閾値はファイル毎に設定が可能である. 自動更新を ON にする事で、自動で閾値を 0.5 下げ、検出処理を繰り返す事が出来る. メディアンフィルタは 3 × 3 の画素で画素値の中央値を取ってから、処理を行うものであり、宇宙線による誤検出を防ぐ事が出来る.

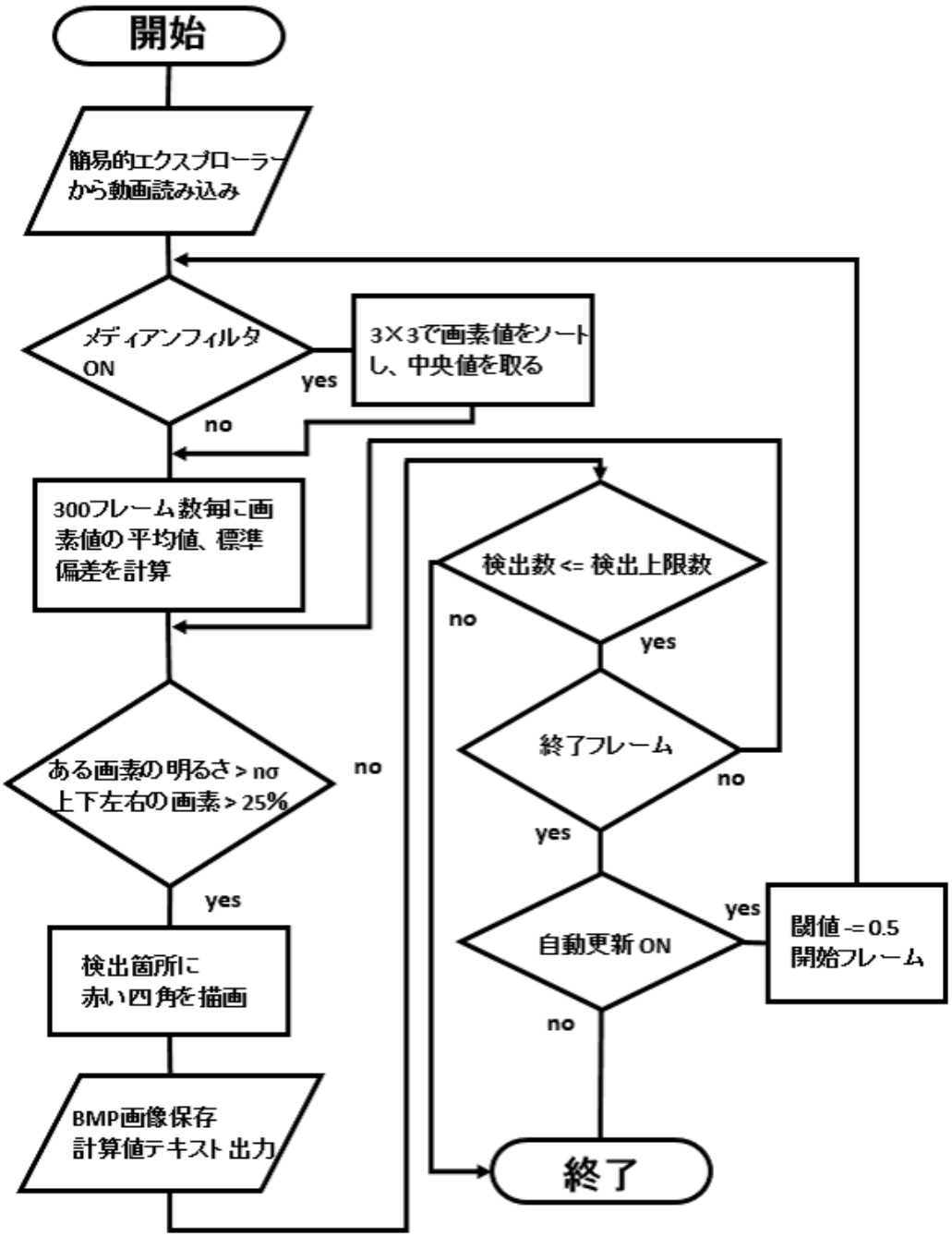

図 2.8: Serdetect のフローチャート図. Serdetect 大まかな処理の流れはこのようになっている. 検出条件の詳しい説明については本文を参照.

2.4.1 検出条件

Serdetect の検出条件は以下のようになっている。まず、指定されたフレーム数 (図 2.7 の処理間隔) 毎に各画素の画素値の平均値、標準偏差 ($\sigma_{x,y}$) を計算する。画素値の平均値をバックグラウンドの明るさとし、画素値からその平均値を引いたものをその画素の明るさとする。ある画素での画素値を $P_{x,y}$、その画素値の平均値を $\overline{P_{x,y}}$ とすれば、その画素の明るさ $I_{x,y}$ は次のように表せる。

$$I_{x,y} = P_{x,y} - \overline{P_{x,y}} \tag{2.1}$$

Sewdetect では以下のような条件を満たすフレームを動画から BMP 形式で保存する。

(1) ある画素で、画素の明るさが σ の n 倍を超える。

$$I_{x,y} \geq n\sigma_{x,y} \tag{2.2}$$

n は図 2.7 の閾値で設定可能な値である。

(2) 更にその画素の明るさとその上下左右の画素の明るさとの比が β を超える。

$$\text{上} : \frac{I_{x,y-1}}{I_{x,y}} \geq \beta \tag{2.3}$$

$$\text{下} : \frac{I_{x,y+1}}{I_{x,y}} \geq \beta \tag{2.4}$$

$$\text{左} : \frac{I_{x-1,y}}{I_{x,y}} \geq \beta \tag{2.5}$$

$$\text{右} : \frac{I_{x+1,y}}{I_{x,y}} \geq \beta \tag{2.6}$$

β は図 2.7 の上下左右の検出条件で設定可能な値であり、今回は $\beta = 0.25$ とした。

第3章　結果

3.1　観測結果

表 2.2 に示すように、2016 年 10 月から 2017 年 1 月の間に行った合計 20.5 時間の観測により、月面閃光のスペクトルと思われる画像が一枚得られた。図 3.1 にそれを示す。日時は 2016/11/07 19:15:54±1(JST) である。Christian Legrand 氏, Patric Chevalley 氏提供のフリーウェア Virtual Moon Atlas を使用して、撮影時の月面の様子と撮影した画像をフィットさせ、位置を調べた (図 3.2)。その結果、月面座標で北緯 20° ± 3 西経 65° ± 5 であった。カメラの設定はゲインが 35 dB(FireCapture 上での設定値は 350)、シャッタースピードが 16.66 ms、ガンマはオフであった。赤い枠線は検出ソフトが 0 次像と思われる位置に描画したものであり、その左側に 1 次像と思われる明線が現れている。 このフレームの前後にスペクトルらしきものは写っていなかった。ただし、ワークステーションの取り込み速度が追いつかず、閃光の寸前の 1 フレームが欠損していた。

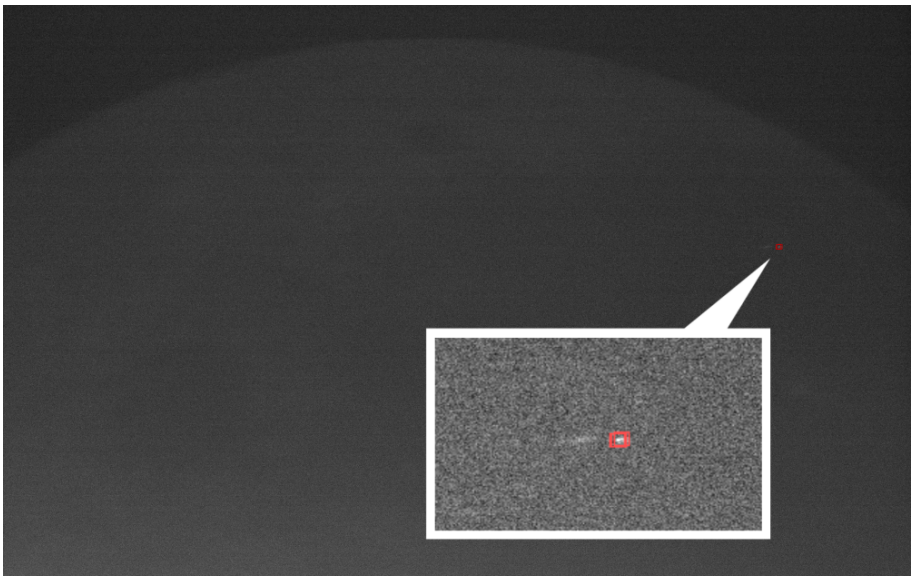

図 3.1: 2016/11/07 19:15:54±1(JST) に撮影した月面閃光と思わしき画像. このフレームの前後にスペクトルらしきものは写っていなかった. ただし、ワークステーションの取り込み速度が追いつかず、閃光の寸前の 1 フレームが欠損していた. 拡大画面内の赤い枠線は検出ソフトが描画したものである.

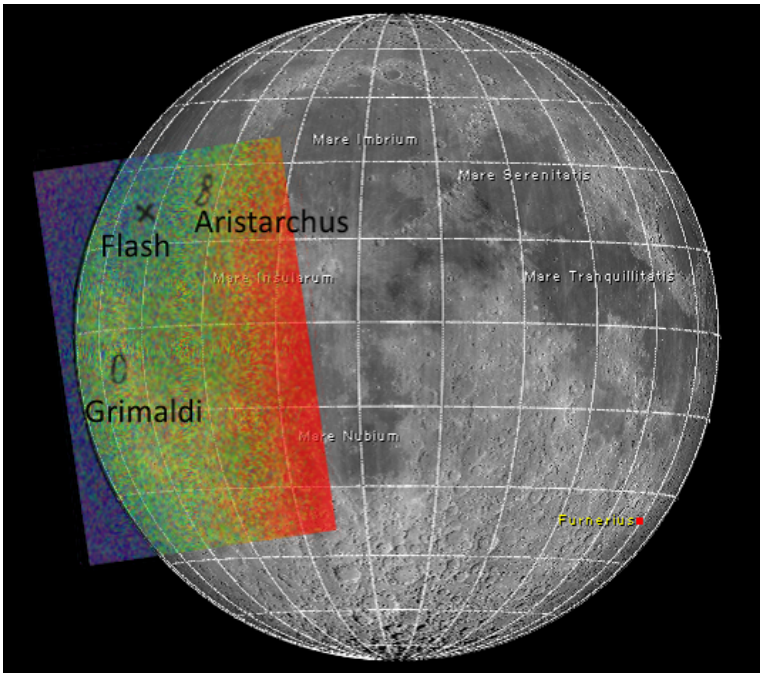

図 3.2: 閃光の位置. フリーウェア Virtual Moon Atlas を使用して、撮影時の月面の様子と閃光の画像をフィットさせた. その結果、位置は月面座標で北緯 20° ± 3、西経 65° ± 5 であった.

第4章　考察

まず、撮影した画像が月面閃光であるかどうかについて議論する。まず、月面閃光と間違われるものとして、人工衛星の映り込みが考えられる。しかし、人工衛星は画面上で、点灯状態または点滅状態で、直線的に移動していくので、月面閃光との区別は基本的に容易である。図 4.1 は月面観測中に撮影した人工衛星と思わしき像である。この像は点灯状態で、直線的に動いていた。図 3.1 の撮影時刻 ±10 秒にそれらしき像は無かった。従って、この撮影した閃光と思わしき画像が、人工衛星である可能性は低いと考えられる。ただし、人工衛星が姿勢を変える過程で一瞬だけ太陽光が反射し、閃光のように写った可能性が無いとは言えない。

次に、この像がスペクトルであるかどうかについて議論する。スペクトルの像に似ているものとして宇宙線によるノイズが考えられる。図 4.2 は Brian Cudnik (2009) が撮影した宇宙線ノイズである。図 4.2 から宇宙線ノイズは直線または曲線のような形状を示す事があると分かる。しかしながら、このカメラではスペクトルが水平方向に伸びるようになっている。撮影した図 3.1 が曲がりくねっていたり、水平方向から大きく傾いている様子が無い点からは、宇宙線ノイズである可能性が低いといえる。更に、撮影した像が可視光スペクトルの範囲で明るく写っていればより一層、宇宙線ノイズである可能性は低いと言えるだろう。図 4.3 は観測した閃光と思わしき画像を拡大し、波長を割り振ったものである。このスペクトルカメラでは可視光領域以外を撮像することは出来ない。従って、もし紫外線の波長範囲 (400 nm 以下) に明るく写っていれば、それはスペクトルでは無い可能性が高くなる。しかし、図 4.3 を見る限り、紫外線の領域に明るく写っているものはない。更に 0 次像と 1 次像と思われる位置で画素値が大きくなっているのが分かる。よって撮影画像から判断する限り、スペクトルである可能性が高いと言えるだろう。

NASA は月面閃光の観測結果を Web 上で随時公開しているが、閃光と思われる画像を撮影した時期のデータは未だ公開されていない。もし、今後のアップデートで、同時刻に月面上の同位置で観測データが挙がれば、撮影した画像は月面閃光である可能性がかなり高くなるだろう。

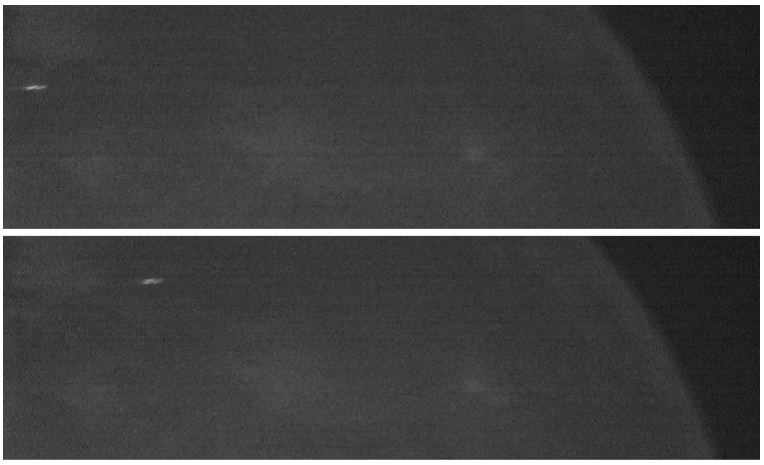

図 4.1: 月面観測中に撮影した人工衛星と思われる像. 上は 2017/01/03 18:03:31±1(JST)、下はその 240 ミリ秒後に撮影した像である. 点灯状態で直線的に動いていた.

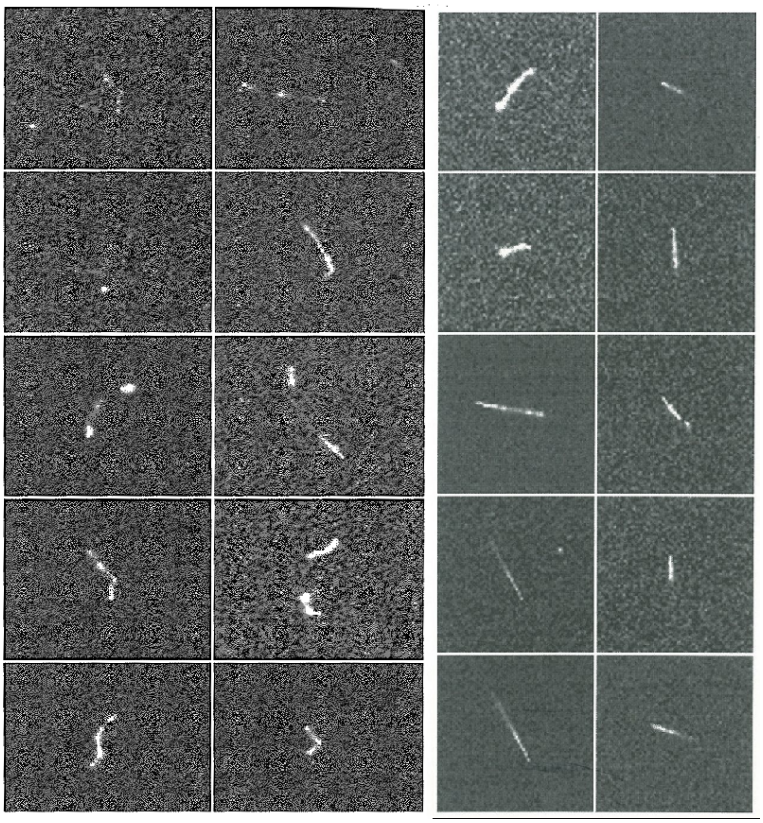

図 4.2: Brian Cudnik 氏が撮影した宇宙線ノイズ [7]. 宇宙線ノイズが直線や曲線のような形状を示す事が分かる.

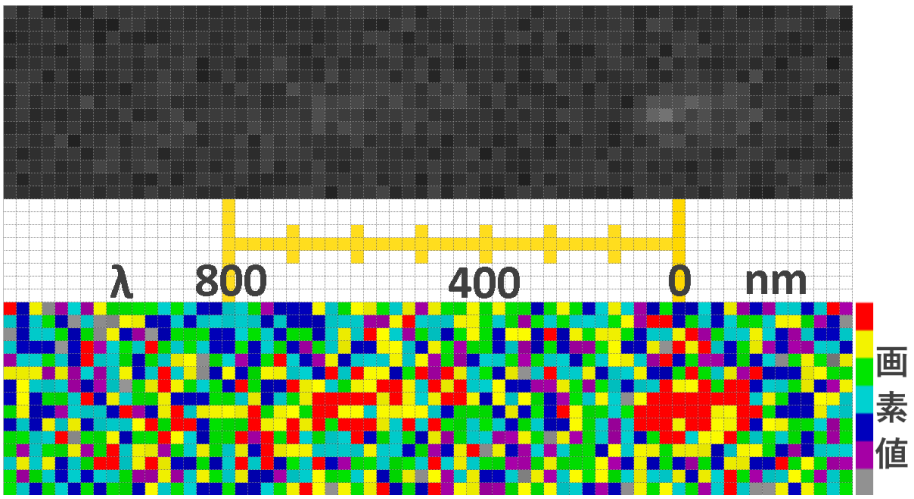

図 4.3: 撮影した閃光と思わしき画像の拡大図. 上の画像は未加工の画像、下の画像は画素値の大きさで色を変えて像を分かりやすくしたものであり、赤色に向かうほど画素値が大きい.

4.1 閃光のスペクトル導出方法

以後は撮影した画像が月面閃光であるとして閃光のスペクトルを導出する。図 4.4 は閃光の分光フラックスと地上で観測される分光フラックスを模式した図である。図 4.4 のように、閃光からの分光フラックスを $F_0(\lambda)$、地上で観測される分光

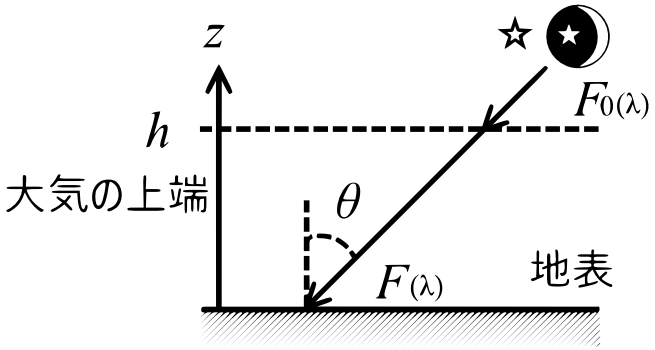

図 4.4: 閃光からの分光フラックス. $F_0(\lambda)$ は閃光の分光フラックス、$F(\lambda)$ は地上で観測されるフラックス、θ は天頂角である.

フラックスを $F(\lambda)$ とすると、$F(\lambda)$ は次のように表すことが出来る。

$$F(\lambda) = F_0(\lambda)e^{-K(\lambda)/\cos\theta} \tag{4.1}$$

ただし、吸収係数を $k(\lambda, z)$ として

$$K(\lambda) = \int_0^h k(\lambda, z)dz \tag{4.2}$$

である。観測されるカウントは $n(\lambda)$ は望遠鏡の口径を D、望遠鏡と回折格子を合わせた透過率を $T(\lambda)$、露光時間 Δt を、波長幅を $\Delta\lambda$、量子効率を $q(\lambda)$、変換係数を p、ゲインを g、A/D 変換を α、プランク定数を h、光速度を c とすると次のように表される。

$$n(\lambda) = F(\lambda) \cdot \pi(\frac{D}{2})^2 \cdot T(\lambda) \cdot \Delta t \cdot \Delta\lambda \cdot \frac{q(\lambda)}{h(\frac{c}{\lambda})} \cdot p \cdot g \cdot \alpha \tag{4.3}$$

カウントを閃光と恒星で比較すると、次式が成り立つ。

$$\frac{n(\lambda)}{n_{☆}(\lambda)} = \frac{F_0(\lambda)e^{-K(\lambda)/\cos\theta}}{F_{0\,☆}(\lambda)e^{-K(\lambda)/\cos\theta_{☆}}} \cdot \frac{\Delta t}{\Delta t_{☆}} \cdot \frac{g}{g_{☆}} \tag{4.4}$$

ただし、同じ望遠鏡と回折格子を使用しているので、

$$D = D_{☆} \tag{4.5}$$
$$T(\lambda) = T_{☆}(\lambda) \tag{4.6}$$

また、同じカメラを使用しているので、

$$\Delta\lambda = \Delta\lambda_{☆} \tag{4.7}$$
$$q(\lambda) = q_{☆}(\lambda) \tag{4.8}$$
$$p = p_{☆} \tag{4.9}$$
$$\alpha = \alpha_{☆} \tag{4.10}$$

と仮定している。
従って、恒星の分光フラックス $F_{0\,☆}(\lambda)$ が分かれば、閃光の分光フラックス $F_0(\lambda)$ は次式で計算できる。

$$F_0(\lambda) = F_{0\,☆}(\lambda) \cdot \frac{\frac{n(\lambda)}{n_{☆}(\lambda)}}{(\frac{\Delta t}{\Delta t_{☆}})(\frac{g}{g_{☆}})} \cdot e^{K(\lambda)(\frac{1}{\cos\theta} - \frac{1}{\cos\theta_{☆}})} \tag{4.11}$$

閃光時の月の天頂角 θ と星を撮像した時のその天頂角 $\theta_{☆}$ が異なる場合には、大気による吸収の違い $e^{K(\lambda)(\frac{1}{\cos\theta} - \frac{1}{\cos\theta_{☆}})}$ を補正する必要がある。なお、$K(\lambda)$ には標準的な大気の天頂方向の透過率のデータを使う。

一方、恒星の分光フラックス $F_{0\,☆}(\lambda)$ は次のようにして求める事が出来る。まず、星の基本情報を調べ、星のスペクトル型 (O, B, A, F, G, K M) から星の有効温度 T を決める。A〜K 型の主系列星では有効温度 T を色指数 $B-V$ を用いて次式で近似できる。

$$T = \frac{9600}{B - V + 1} \tag{4.12}$$

プランクの法則より、プランク定数 h、光速 c、定数 a を用いて $F_{0\,☆}(\lambda)$ は次のように表せる。

$$F_{0\,☆}(\lambda) = a \cdot \frac{2\pi hc^2}{\lambda^5} \frac{1}{e^{\frac{hc}{\lambda hT}} - 1} \tag{4.13}$$

ただし、a の値は次のように決定する。まず、V 等級を決めるためのフィルターの透過率を $f_V(\lambda)$ とし、恒星の見かけの V 等級を m_V、太陽の V 等級を m_S とするとポグソンの式より次式が成り立つ。

$$m_V - m_S = -2.5 \log 10 \left(\frac{\int F_{0\,☆}(\lambda) f_V(\lambda) d\lambda}{\int F_{0S}(\lambda) f_V(\lambda) d\lambda} \right) \tag{4.14}$$

また、式 (4.13) 同様、プランクの法則より、太陽の $F_{0S}(\lambda)$ は太陽の有効温度を T_S として次式が成り立つ。

$$F_{0S}(\lambda) = a_S \cdot \frac{2\pi hc^2}{\lambda^5} \frac{1}{e^{\frac{hc}{\lambda h T_S}} - 1} \tag{4.15}$$

太陽の場合更に、シュテファン・ボルツマンの法則より、太陽定数を S_0 として次の関係が成り立つ。

$$\int_0^\infty F_{0S}(\lambda) d\lambda = a_S \int_0^\infty \frac{2\pi hc^2}{\lambda^5} \frac{1}{e^{\frac{hc}{\lambda h T_S}} - 1} d\lambda = a_s \cdot \sigma T_S^4 = S_0 \tag{4.16}$$

式 (4.16) から a_S を求めることが出来る。さらに式 (4.14) に式 (4.13) と式 (4.15) を代入した式

$$\frac{a \int_0^\infty \frac{2\pi hc^2}{\lambda^5} \frac{1}{e^{\frac{hc}{\lambda h T}} - 1} \cdot f_V(\lambda) d\lambda}{a_S \int_0^\infty \frac{2\pi hc^2}{\lambda^5} \frac{1}{e^{\frac{hc}{\lambda h T_S}} - 1} \cdot f_V(\lambda) d\lambda} = 10^{-\frac{m_V - m_S}{2.5}} \tag{4.17}$$

より、a を求めることが出来る。尚、積分には波長キャリブレーションの結果により、$d\lambda = 20$ nm として数値積分を用いる。

4.2 スペクトル計算結果

まずは恒星の分光フラックス $F_{0\,☆}(\lambda)$ を求める。比較する恒星には、2016/11/7 18:31:13±1(JST) に観測中に写り込んでいた TYC6331-1718-1 (Tycho-2) を用いた。図 4.5 は恒星の実際の様子である。

理科年表 [5, 6] より、$T_S = 5777$ K、$S_0 = 1.37 \times 10^3\,\mathrm{Wm^{-2}}$、$\sigma = 5.67 \times 10^{-8}\,\mathrm{Wm^{-2}K^{-4}}$ を式 (4.16) に代入すると、

$$a_S = \frac{1.37 \times 10^3}{5.67 \times 10^{-8} \cdot 5777^4} = 2.17 \times 10^{-5} \tag{4.18}$$

と求まった。Astro Arts 社 天文シミュレーションソフトウェア ステラナビゲータ 9 より、恒星の色指数 $B - V$ は 1.733 とした。スペクトル型については、情報が見つからなかったため、以後は A 型から K 型のどれかであると仮定し、議論する。式 (4.12) より、恒星の有効温度 $T = 3513$ K と求まった。

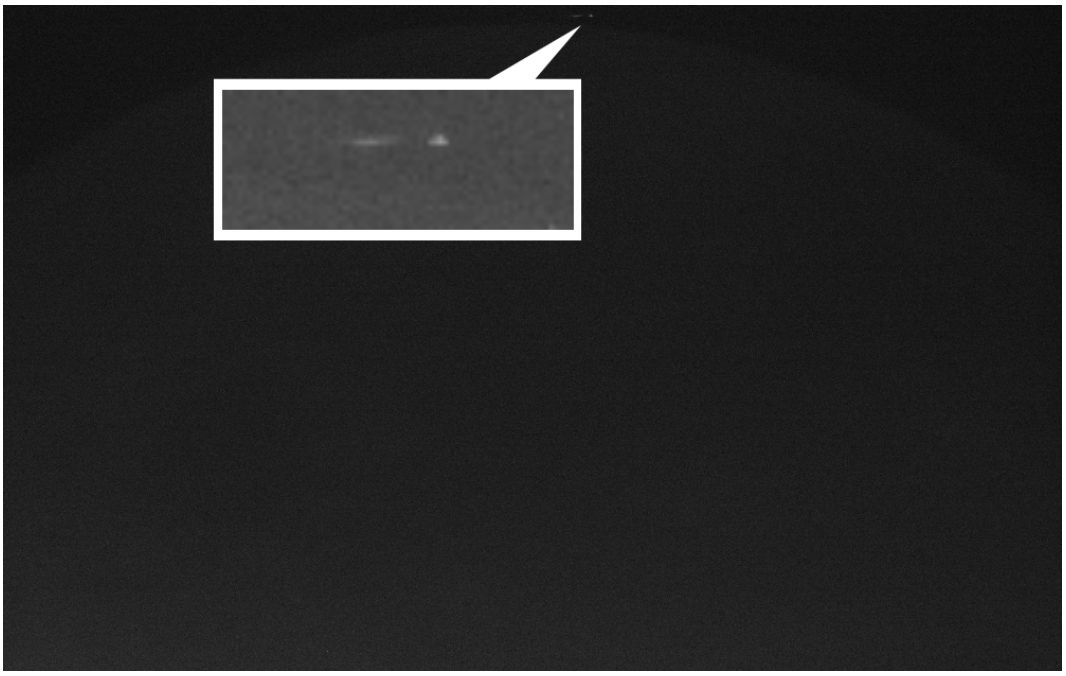

図 4.5: 恒星 TYC6331-1718-1 (Tycho-2). 2016/11/7 18:31:13(JST) に撮影した. カメラの設定は閃光を撮影した際のカメラの設定と等しく、ゲインが 35 dB、シャッタースピードが 16.66 ms であった.

V バンド透過率 $f_V(\lambda)$ の値は Michael S. Bessell (2005) のデータを用いた。理科年表 [6] より、$h = 6.62 \times 10^{-34}$ Js 、$c = 3.00 \times 10^8$ m/s として、式 (4.16) の積分部分を計算したものを表 4.2 に示す。尚、$d\lambda = 20$ nm とした。更に式 (4.17) に、理科年表 [6] より、$m_S = -26.75$、ステラナビゲータ 9 より $m_V = -8.97$ を代入して、

$$a = 2.17 \times 10^{-5} \cdot \frac{7.13 \times 10^6}{3.82 \times 10^5} \cdot 10^{-\frac{8.97-(-26.75)}{2.5}} = 2.09 \times 10^{-18} \tag{4.19}$$

と求まった。この値と式 (4.13) より求めた恒星の分光フラックス $F_{0\,☆}(\lambda)$ [W/m^2/m] を表 4.2 に示す。

最後に、閃光のフラックス $F_0(\lambda)$ を求める。撮影時のカメラの設定は閃光を撮影した際のカメラの設定と等しく、ゲインが 35 dB、シャッタースピードが 16.66 ms であった。従って、式 (4.11) の分母は無視できる。大気透過率は G. Schubert *et al.* (1999) のデータを用いた。ステラナビゲータ 9 で天頂角を調べたところ、$\theta = 58°$、$\theta_☆ = 54°$ であった。カウントは波長毎に、スペクトルが写っている箇所の画素値を列毎に足し、バックグラウンドを引く事によって求めた。閃光のカウント n のバックグラウンドには閃光前の 100 フレームを平均した画素値を用いた。恒星のカウント $n_☆$ のバックグラウンドには周囲の 10 画素を用いた。式 (4.11) を用いて求めた閃光の分光フラックス $F_0(\lambda)$ [fW/m^2/nm] を表 4.2 に示す。

表 4.1: 恒星のフラックス計算.

波長 (nm)	V バンド透過率 $f_V(\lambda)$	$\frac{2\pi hc^2}{\lambda^5}\frac{1}{e^{\frac{hc}{\lambda hT}}-1}\cdot f_V(\lambda)$	$\frac{2\pi hc^2}{\lambda^5}\frac{1}{e^{\frac{hc}{\lambda hT_S}}-1}\cdot f_V(\lambda)$
800	0.00	0.00.E+00	0.00.E+00
780	0.00	0.00.E+00	0.00.E+00
760	0.00	0.00.E+00	0.00.E+00
740	0.00	0.00.E+00	0.00.E+00
720	0.00	0.00.E+00	0.00.E+00
700	0.00	0.00.E+00	0.00.E+00
680	0.01	6.25.E+10	6.78.E+11
660	0.02	1.21.E+11	1.40.E+12
640	0.08	4.64.E+11	5.81.E+12
620	0.19	1.05.E+12	1.42.E+13
600	0.36	1.88.E+12	2.77.E+13
580	0.57	2.79.E+12	4.50.E+13
560	0.78	3.53.E+12	6.28.E+13
540	0.97	4.02.E+12	7.93.E+13
520	0.96	3.59.E+12	7.92.E+13
500	0.46	1.53.E+12	3.81.E+13
480	0.03	8.67.E+10	2.47.E+12
460	0.00	0.00.E+00	0.00.E+00
440	0.00	0.00.E+00	0.00.E+00
420	0.00	0.00.E+00	0.00.E+00
400	0.00	0.00.E+00	0.00.E+00
積分値		3.82.E+05	7.13.E+06

表 4.2: 閃光の分光フラックス.

波長 (nm)	閃光のカウント n	恒星のカウント $n_{\star}$	大気透過率 $e^{-\mathrm{K}(\lambda)}$	$n/n_{\star}$	恒星のフラックス $F_{0\,\star}$ [W/m^2/m]	閃光のフラックス F_0 [fW/m^2/nm]
800	185	759	0.767	0.24	1.4.E-05	3.7
780	393	983	0.755	0.40	1.4.E-05	6.0
760	187	1149	0.744	0.16	1.4.E-05	2.4
740	516	1227	0.732	0.42	1.4.E-05	6.2
720	421	1303	0.721	0.32	1.4.E-05	4.7
700	679	1353	0.709	0.50	1.3.E-05	7.2
680	207	1487	0.705	0.14	1.3.E-05	1.9
660	780	1511	0.701	0.52	1.3.E-05	7.0
640	416	1588	0.691	0.26	1.2.E-05	3.4
620	448	1503	0.675	0.30	1.2.E-05	3.7
600	707	1480	0.659	0.48	1.1.E-05	5.6
580	494	1225	0.650	0.40	1.0.E-05	4.5
560	592	1581	0.641	0.37	9.4.E-06	3.8
540	557	1521	0.628	0.37	8.6.E-06	3.5
520	498	1363	0.609	0.37	7.8.E-06	3.1
500	322	1226	0.591	0.26	6.9.E-06	2.0
480	277	992	0.565	0.28	6.0.E-06	1.9
460	354	778	0.540	0.45	5.1.E-06	2.6
440	393	722	0.509	0.54	4.3.E-06	2.7
420	222	396	0.474	0.56	3.5.E-06	2.2
400	52	285	0.438	0.18	2.7.E-06	0.6

図 4.6 は閃光のフラックス [fW/m^2/nm] を波長毎に表したグラフである。図 4.6 の誤差は次のようにして求めた。まず、閃光画像の画素値が閃光前の 100 フレーム同等にばらつきを持っていると仮定し、各画素で 100 フレームの標準偏差を求めた。閃光の写り込んだ範囲で標準偏差の平均値を求めると 60.1 であった。この値を各画素における画素値の誤差とした。

各画素における画素値の誤差を δ_p とすれば、n は 10 画素を合計して求めていたので、n の誤差 δ_n は次式のように表せる。

$$\delta_n = \delta_p\sqrt{10} \tag{4.20}$$

一方、$n_☆$ の誤差 $\delta_{n_☆}$ は、30 フレームで平均し、11 画素を合計し、バックグラウンドとして 10 画素を平均し、合計から引いたので、次式のよう表せる。

$$\delta_{n_☆} = \sqrt{\left(\sqrt{\frac{11}{30}}\delta_p\right)^2 + \left(\frac{11\delta_p}{\sqrt{30}\cdot\sqrt{10}}\right)^2} = \delta_p\sqrt{\frac{231}{300}} \tag{4.21}$$

δ_n と $\delta_{n_☆}$ を用いれば、$n/n_☆$ の誤差 $\delta_{n/n_☆}$ は次式で表すことが出来る。

$$\delta_{n/n_☆} = \sqrt{\left(\frac{\delta_n}{n_☆}\right)^2 + \left(\frac{n}{n_☆^2}\delta_{n_☆}\right)^2} \tag{4.22}$$

あとは、同様の方法でフラックスを求め、フラックスの誤差とした。

図 4.6 の点線はプランクの法則を用いて求めた黒体放射スペクトルを定数倍して閃光のスペクトルにフィットさせたものである。誤差範囲を含め、閃光のスペクトルに近いものは 2800 K から 4800 K における黒体放射スペクトルであった。Nemtchinov *et al.* (1998) によれば、10 cm サイズのメテオロイドが、高度 30°、速度 25 km/s で天体に衝突した際に、2800 K の蒸気雲または液滴が発生すると推測されている [8]。実際、図 4.6 に示すように、誤差範囲を含めれば、閃光のスペクトルは 2800 K における黒体放射スペクトルに近いスペクトルになった。

以下は比較に用いた恒星の温度について議論する。表 4.3 は主系列星における温度と色指数等の関係を表したものである。色指数は基本的に天体の表面温度が低いほど値が大きくなる。ステラナビゲータ 9 によれば、今回用いた比較星の色指数 $B-V$ は 1.733 であった。従って、今回用いた比較星は近似式 (4.12) が成り立たない M 型星である可能性がある。表 4.3 から分かるように、M 型星の有効温度を近似式 (4.12) で求めようとした場合、実際の値よりも数百度高くなってしまう事が考えられる。従って、実際よりも恒星の温度を大きく見積もってしまった可能性を考慮し、温度を 3000 K、2500 K まで引き下げた際に閃光のフラックスがどうなるか計算したものを、図 4.7 に示す。その結果、スペクトルの右肩上がりな形が、より顕著になり Nemtchinov *et al.* (1998) によって推定された 2800 K における黒体放射スペクトルにより近い形となった。

今回求めた閃光のスペクトルは総じて、可視光領域で高波長にかけて右肩上がりの形を示した。よって、赤色付近の波長範囲で感度の高い機材を使用すること

で、今後の観測が容易になると考えられる。一方、誤差の範囲が大きく、今回の環境では、正確な閃光の温度を求めることは難しいことが分かった。誤差の値は式 (4.22) から分かるように、恒星画像のカウントよりも、閃光画像のカウントによる影響が大きい。そのため、スペクトルをより正確に測定するためには、撮影時のカメラのゲインを下げ、一般的な月面閃光よりも明るい閃光に的を絞って観測をするのが良いだろう。ゲインを下げる事により各画素での誤差が小さくなる。それだけでなく、規模の大きい閃光の場合、複数のフレームに渡って映るので、画素値を平均する事によって更に誤差が小さくなるだろう。

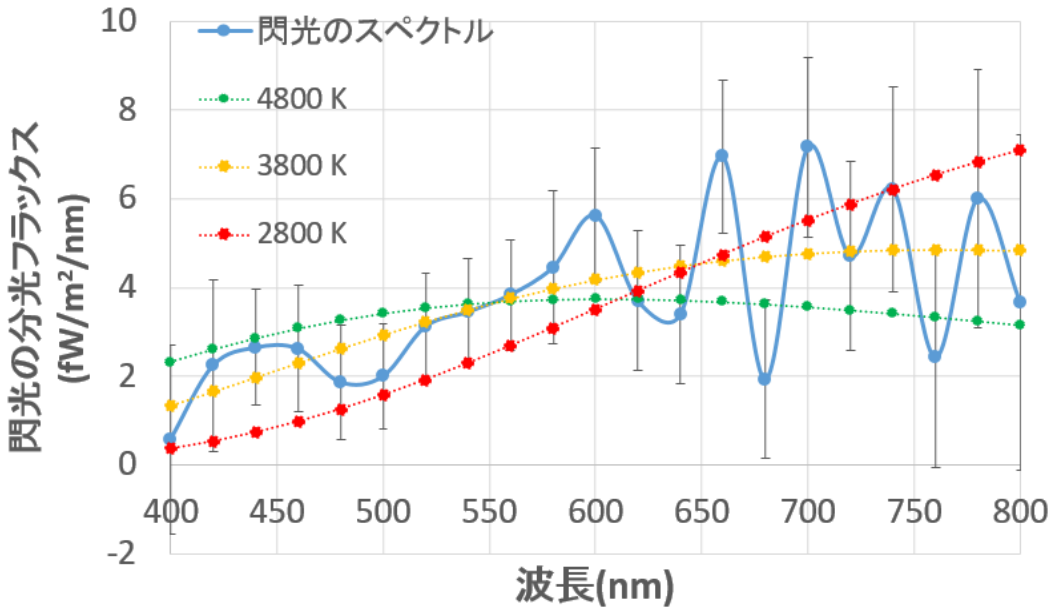

図 4.6: 閃光のスペクトル．縦軸を閃光のフラックス (fW/m^2/nm)、横軸を波長 (nm) として閃光のスペクトルを求めた．青線は閃光のあった日に月の近くに写り込んでいた恒星を用いて求めた閃光のスペクトルである．点線は 4800 K, 3800 K, 2800 K における黒体放射スペクトルを定数倍し、閃光のスペクトルにフィットさせたものである．

表 4.3: 主系列星におけるスペクトル型と有効温度等の関係 [9].

スペクトル型	有効温度 (K)	色指数(等) $B-V$	色指数(等) $U-B$	実視絶対等級(等)	全放射補正 (等)
O5	45000	−0.3	−1.1	−5.5	−4
B0	29000	−0.3	−1.1	−4	−2.8
B5	15000	−0.16	−0.56	−1	−1.3
A0	9600	0.00	0.00	0.5	−0.2
A5	8300	0.15	0.11	1.8	0
F0	7200	0.33	0.03	2.4	0
F5	6600	0.45	0.00	3.2	0
G0	6000	0.60	0.12	4.4	−0.1
G5	5600	0.68	0.23	5.1	−0.1
K0	5300	0.81	0.46	5.9	−0.2
K5	4400	1.15	1.1	7.2	−0.6
M0	3900	1.4	1.2	8.7	−1.2
M5	3300	1.6	1.2	12	−2.4

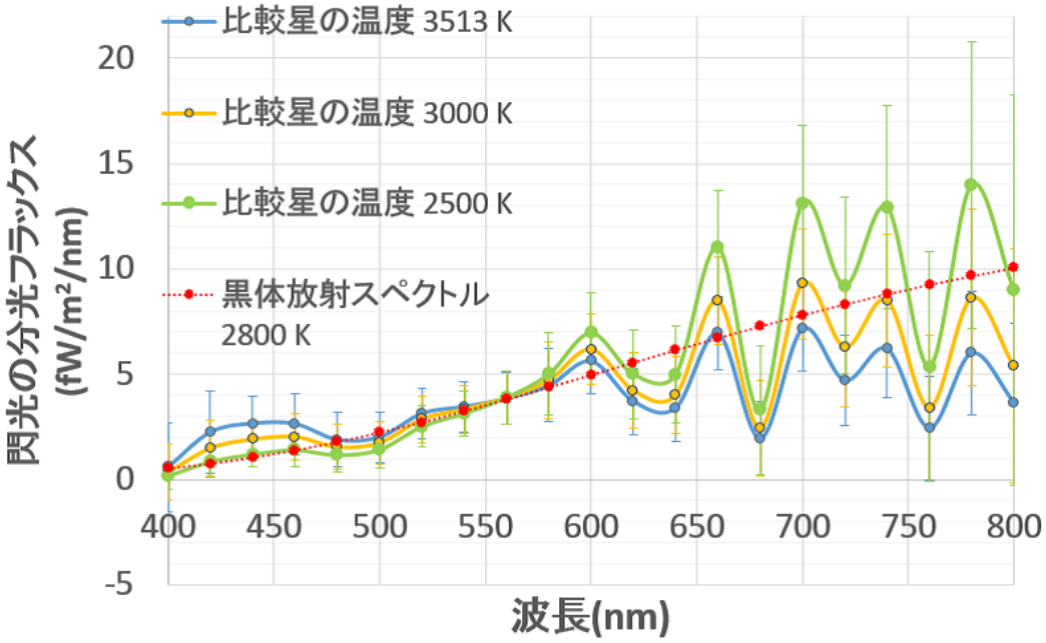

図 4.7: 比較星の温度を変化させた時の閃光のスペクトル．縦軸を閃光のフラックス (fW/m^2/nm)、横軸を波長 (nm) として閃光のスペクトルを求めた．Nemtchinov *et al.* (1998) の推定温度に従い、点線で 2800 K における黒体放射スペクトルを定数倍したものを添えた．

謝辞

3年間の研究生活において、丁寧にご指導して頂いた柳澤正久教授に深く感謝致します。観測を手伝って下さった島田君や内田さん、石川君、ありがとうございました。そして、東畑班を手伝っていた小林君、栗原君、山本君、お疲れ様でした。

参考文献

[1] R. M. Suggs, D. E. Moser, W. J. Cooke, R. J. Suggs, “The flux of kilogram-sized meteoroids from lunar impact monitoring”, Icarus, vol. 238, p. 23, 2014.

[2] 池上 裕美, 平成 19 年度修士論文 “2007 年ふたご座流星群による月面衝突閃光の観測”, 電気通信大学, p. 15, 2008.

[3] Michael S. Bessell, “Standard Photometric Systems”, Annual Review of Astronomy and Astrophysics, vol. 43, p. 304, 2005.

[4] G. Schubert, R. L. Walterschied 1999. Earth In: A. N. Cox (Ed), Allen’s Astrophysical Quantities, 4th ed. Springer, New York, pp. 239-292.

[5] 自然科学研究機構 国立天文台, “理科年表プレミアム: 太陽の諸定数”, http://www.rikanenpyo.jp/member/?module=Member&action=Contents&page=allASx31x0040_2017_1.html, (参照 2017-1-27).

[6] 自然科学研究機構 国立天文台, “理科年表プレミアム: 基礎物理定数”, http://www.rikanenpyo.jp/member/?module=Member&action=Contents&page=allPSx31x0110_2017_1.html, (参照 2017-1-27).

[7] Brian Cudnik, “Lunar Meteoroids Impacts and How to Observe Them”, Springer, pp. 173-174, 2009.

[8] I. V. Nemtchinov, V. V. Shuvalov, N. A. Artemieva, B. A. Ivanov, I. B. Kosarev, I. A. Trubetskaya, “Light Impulse Created by Meteoroids Impacting the Moon”, Lunar and Planetary Science XXIX, Abstract 1032, 1998.

[9] 吉岡 一男, “天体物理学入門”, 放送大学教育振興会, p. 132, 2003.

付録

Serdetect.exe ソースコード

main.cpp

```
#include "header.h"

int ImageWidth, ImageHeight, PixelDepthPerPlane, FrameCount;
int ImageWidth2, ImageHeight2, ImageSize; //画像の幅と高さの 1/2、および 1フレームの全画素数

uint16 *value16;     //16it_ser 画素値の画像全体分の配列
uint16 *value16_tmp;
uint8 *value8;      // 8it_ser 画素値の画像全体分の配列
double *x2heikin; //平均計算のために輝度値の合計を入れる
double *ave;
double *sigma; //平均値を入れる
//char *fname;
//////////////////////////////////////////リストビュー
LPCTSTR szBuf_edit1;
LPTSTR strItem0[] = { TEXT("ファイル名"), TEXT("開始"), TEXT("終了"), TEXT("上限"), TEXT("閾値") };
int CX[] = { 80,40,40,40,40 };//コラムの横幅
char slash[2] = {"\\"};
////////////////////////////////////////////////
HWND hImage1;

char szClassName0[] = "img_data0"; //全体表示用ウィンドウクラスの名
LRESULT hr;
char szBuf[256];
int xxx, threshold, count, detect, number, txtfilenum, camera_no, date;
double xx;
int frame[120000], detect_i[120000], detect_j[120000], kido[120000];

HWND hDlg, hImage0; //ダイアログウィンドウのハンドル
//HWND hRadio1, hRadio2, hRadio3; //ラジオボタン用のハンドルを準備する
HWND hRadio_median_on, hRadio_median_off;
HWND hRadio_auto_update_on,hRadio_auto_update_off;
HWND hCheck1, hCheck2; //チェックボタン用のハンドルを準備する

IMG0 g_img; //IMG0 はヘッダーで定義されている
POINTS mp0, mp1;  //マウスの座標用

int WINAPI WinMain( HINSTANCE hCurInst, HINSTANCE hPrevInst,
                                        LPSTR lpszCmdLine, int nCmdShow )
{
        MSG msg;
        BOOL bRet;
        g_img.hi0 = hCurInst;
        g_img.hwin = NULL;
        g_img.lpBmpData = NULL;

        number = txtfilenum = 0;
        if (!InitApp0(hCurInst)) //全体表示用ウィンドウクラスを登録する
                return FALSE;
        hDlg = CreateDialog( hCurInst, "IDD_DIALOG1", HWND_DESKTOP, //ダイアログボックスを作る
                                        (DLGPROC)dlgproc);
        while ((bRet = GetMessage(&msg, NULL, 0, 0)) != 0) {
                if (bRet == -1) {
                        break;
                } else {
                        if (!hDlg || !IsDialogMessage(hDlg, &msg)) {
                                TranslateMessage(&msg);
                                DispatchMessage(&msg);
                        }
                }
        }

        free( g_img.lpBmpData );
        return TRUE;
```

```
64  }
65
66  //全体画面用ウィンドウクラスの登録
67  ATOM InitApp0( HINSTANCE hInst )
68  {
69          WNDCLASSEX wc0;     //新しくつくるウインドウクラス
70          memset( &wc0, 0, sizeof(WNDCLASSEX));
71          wc0.cbSize = sizeof(WNDCLASSEX);
72          wc0.lpfnWndProc = WndProc0;  //このクラスの持つウインドプロシージャ
73          wc0.hInstance = hInst;
74          wc0.hbrBackground = (HBRUSH)GetStockObject( WHITE_BRUSH );
75          wc0.lpszClassName = (LPCSTR)szClassName0; //このクラスの名前
76          return (RegisterClassEx( &wc0 ));
77  }
78
79  //全体画面用ウィンドウの作成
80  BOOL InitInstance0(HINSTANCE hInst, int nCmdShow)
81  {
82          int y_cap = GetSystemMetrics(SM_CYCAPTION);
83          int y_frm = GetSystemMetrics(SM_CYFIXEDFRAME);
84          int x_frm = GetSystemMetrics(SM_CXFIXEDFRAME);
85          RECT rect;
86          //MessageBox(NULL,"CreateWindow 開始","debug",MB_OK);
87          //全体表示用ウィンドウの作成
88          if(GetWindowRect(hDlg, &rect) == FALSE){//ダイアログボックスの位置を取得
89                  MessageBox(NULL, "ウィンドウの位置が特定できない","debug", MB_OK);
90                  return TRUE;
91          }
92
93          g_img.hwin = CreateWindow( (LPCSTR)szClassName0, // クラスの名前
94                                                           "SER 画像",    // キャプション
95                                                           WS_OVERLAPPED | WS_VISIBLE,  // ウィンドウの属性
96                                                           rect.left - ImageWidth2, rect.top,//ダイアログボックスの左にぴったり付ける
97                                                           ImageWidth2 + y_frm * 2 + 10,
98                                                           ImageHeight2 + y_cap + x_frm * 2 + 10,
99                                                           NULL,
100                                                          NULL, hInst, NULL );
101
102         if(!g_img.hwin){
103                 MessageBox(NULL,"bug","debug",MB_OK);
104                 return FALSE;
105         }
106
107         ShowWindow(g_img.hwin, nCmdShow);
108         UpdateWindow(g_img.hwin);
109         return TRUE;
110 }
111
112 //様々な作業の中心となるダイアログ
113 BOOL CALLBACK dlgproc( HWND hDlg, UINT msg, WPARAM wp, LPARAM lp )
114 {
115         //フォルダを開く動作用↓
116         BROWSEINFO bi;
117         ITEMIDLIST *idl;    //__unaligned この修飾子は x64 アプリケーションでは必要ありません。
118         LPMALLOC g_pMalloc;
119         char szTmp[MAX_PATH]={ "C:\\*.ser" };//フォルダパス
120         SHGetMalloc(&g_pMalloc);  //インタフェースを取得
121         bi.hwndOwner = NULL;
122         bi.pidlRoot = NULL;
123         bi.pszDisplayName = szTmp;
124         bi.lpszTitle = TEXT("フォルダを選択してください");
125         bi.ulFlags = BIF_RETURNONLYFSDIRS;
126         bi.lpfn = NULL;
127         bi.lParam = 0;
128         bi.iImage = 0;
129
130         //リストビュー操作用↓
131         static DLIST listctrl;   //検出リストの情報を保存する.
132         LVCOLUMN lvcol;
133     LVITEM list_item;
134     UINT ic;
135     static HWND hList;
136         static HWND hEdit;
137     //LV_DISPINFO *lvinfo;
138     //TCHAR buf[64];
139     DWORD dwStyle;
140         static int add_num = 0;   //検出リストのアイテム数
141         static errno_t error;
142         UINT uiState;     //検出リストの選択状況
143
144         //その他↓
145         LPNMUPDOWN lpnmud;    //スピンコントロール用
146         static int mag_brt;    //全体画面の明るさ
147         //static RECT cur_rect;   //ウィンドウの現在の座標
148         static RECT dlg_rect;   //ダイアログボックスの座標
149         const int headersize = 178;
```

```
150         static char *fname;    //ファイル名
151         static FILE *fp;
152         //WCHAR filename[ MAX_PATH ]; //WCHAR に変換したファイル名
153         static int max_frame;
154         static int c_frame;   //フレーム番号
155
156         int i, j, l, n, hani, repeat, hidari, migi, ue, shita, end, square, allframe, filesave;
157         static int Cur_ImageSize;
158         //int k;
159         static BOOL hist_on = FALSE;
160         static int last_hani;
161         static char *folder_name;
162         char ch[100]={};
163         char detect_dir[100]={};   //検出フォルダー名//Detection_Moon_xxxxxx
164         char detect_result_path[100]={}; //検出ログ用パス//Detection_Moon_xxxxxx\\SER_Detection_Result.txt
165         char detect_dir_c_frame[100]={}; //検出後フォルダー名変更用
                //Detection_Moon_xxxxxx_c_frame//検出フォルダー名+最後のフレーム
166         static LONGLONG nDuration = 0;  //ファイル中の全フレーム数
167         static int detect_limit;   //検出上限フレーム数
168         BOOL check;       //色々に使う
169         //static char szTmp[255]= { "C:\\*.*" };
170         switch(msg){
171         case WM_INITDIALOG:
172                 //リストビュー操作用↓
173                 SendMessage(GetDlgItem(hDlg, IDC_SPIN1), UDM_SETBUDDY,
174                                 (WPARAM)GetDlgItem(hDlg, IDC_EDIT_MAG_BRT), 0);//スピンコントロール
                                        と隣のエディットボックスを関連付ける.
175                 InitCommonControls();
176         hList = GetDlgItem(hDlg, IDC_LIST2);
177         dwStyle = ListView_GetExtendedListViewStyle(hList);
178         dwStyle |= /*LVS_EX_CHECKBOXES*/ LVS_EX_FULLROWSELECT |//チェックボックスをつけるならコメントア
                ウト解除
179                 LVS_EX_GRIDLINES | LVS_EX_HEADERDRAGDROP;
180         ListView_SetExtendedListViewStyle(hList, dwStyle);
181
182         lvcol.mask = LVCF_FMT | LVCF_WIDTH | LVCF_TEXT | LVCF_SUBITEM;
183         lvcol.fmt = LVCFMT_LEFT;
184         for(ic = 0; ic < 5; ic++)   //コラムの数を増やす
185         {
186             lvcol.cx = CX[ic];    // 表示位置
187             lvcol.pszText = strItem0[ic]; // 見出し
188             lvcol.iSubItem = ic;   // サブアイテムの番号
189             ListView_InsertColumn(hList, ic, &lvcol);
190         }
191
192         list_item.mask = LVIF_TEXT;
193
194                 //エディットボックス等に初期値を設定
195                 hRadio_median_on = GetDlgItem(hDlg, IDC_RADIO_MEDIAN_ON); //ラジオボタンのハンドルを取得
196                 hRadio_median_off = GetDlgItem(hDlg, IDC_RADIO_MEDIAN_OFF);
197                 Button_SetCheck(hRadio_median_off, BST_CHECKED); //デフォルトではメディアンフィルターオフに設定
198                 hRadio_auto_update_on = GetDlgItem(hDlg, IDC_RADIO_AUTO_UPDATE_ON);
199                 hRadio_auto_update_off = GetDlgItem(hDlg, IDC_RADIO_AUTO_UPDATE_OFF);
200                 Button_SetCheck(hRadio_auto_update_on, BST_CHECKED);//デフォルトでは自動更新オンに設定
201                 SetDlgItemInt( hDlg, IDC_EDITthreshold, 50, FALSE ); //検出条件をデフォルトで 5に設定
202                 SetDlgItemInt( hDlg, IDC_EDIThani, 300, FALSE ); //検出フレーム数をデフォルトで 300に設定
203                 SetDlgItemInt( hDlg, IDC_EDITue, 25, FALSE );
204                 SetDlgItemInt( hDlg, IDC_EDITshita, 25, FALSE );
205                 SetDlgItemInt( hDlg, IDC_EDITmigi, 25, FALSE );
206                 SetDlgItemInt( hDlg, IDC_EDIThidari, 25, FALSE );
207                 SetDlgItemInt( hDlg, IDC_EDIT_MAG_BRT, 200, FALSE );
208                 SetDlgItemInt( hDlg, IDC_EDIT_DETECT_LIMIT, 200, FALSE );//デフォルトでは検出上限フレーム数を
                        200に設定
209                 DlgDirList(hDlg,szTmp,IDC_LIST1,IDC_EDIT_PATH,DDL_DIRECTORY | DDL_DRIVES );//フォル
                        ダ内の SER ファイルをリストに表示
210                 return TRUE;
211         case WM_NOTIFY:
212                 //検出リスト内のアイテム編集.//クリックしたときの動作
213                 hList = GetDlgItem(hDlg, IDC_LIST2);
214                 if(add_num > 0){
215                         for(i = 0; i < add_num; i++){
216                                 uiState = ListView_GetItemState(hList, i,LVIS_SELECTED | LVIS_FOCUSED);
217                                 if ( uiState & LVIS_SELECTED ){ // 選択された時､閾値などの値をそれぞれのエディットボ
                                        ックスにセット.
218                                   SetDlgItemInt(hDlg, IDC_EDITstart,   listctrl.start_no[ add_num - i -1],
                                          FALSE);
219                                   SetDlgItemInt(hDlg, IDC_EDITend,   listctrl.end_no[ add_num - i -1],
                                          FALSE);
220                                   SetDlgItemInt(hDlg, IDC_EDIT_DETECT_LIMIT, listctrl.limit_no[
                                          add_num - i -1], FALSE);
221                                   SetDlgItemInt(hDlg, IDC_EDITthreshold,  listctrl.threshold[ add_num - i
                                          -1], FALSE);
222                                   SetDlgItemInt(hDlg, IDC_EDITallframe,  listctrl.allframe[ add_num - i
                                          -1], FALSE);
223                                   break;
224                                 }
225
226                         }
227                 }
```

```
228                     //全体画面の明るさをスピンコントロールで調整する
229                     if (wp == (WPARAM)IDC_SPIN1){
230                             lpnmud = (LPNMUPDOWN)lp;
231                             if(lpnmud->hdr.code == UDN_DELTAPOS){
232                                     mag_brt = GetDlgItemInt(hDlg, IDC_EDIT_MAG_BRT, &check, FALSE);
233
234                                     if((lpnmud->iDelta) < 0)
235                                     {
236                     mag_brt += 50;
237                                     }
238                                     else if((lpnmud->iDelta) > 0)
239                                     {
240                     mag_brt -= 50;
241                                     }
242                                     if(mag_brt > 0){//0以下になってしまうなら変更しない.
243                                             SetDlgItemInt(hDlg, IDC_EDIT_MAG_BRT,mag_brt, FALSE);
244                                             SendMessage (hDlg, WM_COMMAND, MAKEWPARAM(
                                                    IDC_BUTTON_FRAME, 0), 0);
245                                     }
246                             }
247                     }
248                     break;
249
250     case WM_COMMAND:
251     switch(LOWORD(wp)){
252     case IDC_BUTTON_FOLDER_OPEN://フォルダを開く.
253             //ダイアログを表示
254             idl = SHBrowseForFolder(&bi);
255             if(idl != NULL)
256             {
257                     if(SHGetPathFromIDList(idl,szTmp) != FALSE){
258                             strcat_s(szTmp,"\\*.ser");//SER ファイルのみ表示
259                             //MessageBox(NULL,szTmp,TEXT("フォルダパス"),MB_OK);
260                             DlgDirList(hDlg,szTmp,IDC_LIST1,IDC_EDIT_PATH,DDL_DIRECTORY | DDL_DRIVES);
261                     }
262                     //strcat_s(szTmp,"\\*.ser");//SER 形式のみ表示する場合は DDL_READWRITE | DDL_READONLY のみ
263
264                     //PIDL を解放する
265                     //g_pMalloc->lpVtbl->Free(g_pMalloc,idl); //for c
266                     g_pMalloc->Free(idl); //for c++//インタフェース開放
267             }
268             return TRUE;
269     case IDC_LIST1:    //簡易エクスプローラー
            (左リスト)からファイルを指定する//http://www.eonet.ne.jp/~maeda/winc/dirlist.htm 参照
270             //MessageBox(NULL,"IDC_LIST1","debug", MB_OK);
271             if(HIWORD(wp) == LBN_DBLCLK){           //ダブルクリックした時、カレントディレクトリを移動
272                     if (DlgDirSelectEx(hDlg,szTmp,sizeof(szTmp),IDC_LIST1)){
273                             strcat_s(szTmp,"*.ser");//SER ファイルのみ表示
274                             //MessageBox(NULL,szTmp,TEXT("フォルダパス"),MB_OK);
275                             DlgDirList(hDlg,szTmp,IDC_LIST1,IDC_EDIT_PATH,DDL_DIRECTORY | DDL_DRIVES);
276                     }else{
277                             //MessageBox(hDlg,szTmp,"File Selected",MB_OK | MB_ICONINFORMATION);
278                             SendMessage (hDlg, WM_COMMAND, MAKEWPARAM(IDC_BUTTON_ADD, 0), 0); //
                                    ファイルを追加
                                    する
279                     }
280             }
281             break;
282     case IDC_BUTTON_LISTHIST: //リスト検出ボタン.
283             /*検出するファイルの確認用
284             char ctes[2000];
285             char buf[100];
286             wsprintf(ctes, "");
287             wsprintf(buf, "");
288             for(i=0; i< add_num; i++){
289
290                     strcat_s(ctes, "No.");
291                     wsprintf(buf, "%d_", i);
292                     strcat_s(ctes, buf);
293                     strcat_s(ctes, listctrl.file_path[i]);
294                     //strcat_s(ctes, "\n");
295                     strcat_s(ctes, "_");
296                     strcat_s(ctes, "開始:");
297                     wsprintf(buf, "%d_", listctrl.start_no[i]);
298                     strcat_s(ctes, buf);
299
300                     strcat_s(ctes, "終了:");
301                     wsprintf(buf, "%d_", listctrl.end_no[i]);
302                     strcat_s(ctes, buf);
303
304                     strcat_s(ctes, "上限:");
305                     wsprintf(buf, "%d_", listctrl.limit_no[i]);
306                     strcat_s(ctes, buf);
307
308                     strcat_s(ctes, "閾値:");
309                     wsprintf(buf, "%d_", listctrl.threshold[i]);
310                     strcat_s(ctes, buf);
311
```

```
312                     strcat_s(ctes, "総フレーム数:");
313                     wsprintf(buf, "%d\n", listctrl.allframe[i]);
314                     strcat_s(ctes, buf);
315             }
316             MessageBox(NULL, ctes, "検出処理を行うファイル一覧", MB_OK);
317             return 0;
318             */
319             for(i=0; i< add_num; i++){
320                     SetWindowText(g_img.hwin,listctrl.file_path[i]);//ウィンドウの名前を変更.
321                     //MessageBox(NULL, listctrl.file_path[i], "このファイルを検出します。", MB_OK );
322                     //wsprintf(ctes,"start=%d, end=%d",listctrl.start_no[i],listctrl.end_no[i]);
323                     //MessageBox(NULL, ctes, "このファイルを検出します。", MB_OK );
324                     fname = listctrl.file_path[i];
325                     wsprintf(ch, TEXT("%s"), fname);
326                     folder_name = PathFindFileName(ch);
327                     PathRemoveExtension(folder_name);
328                     //Tech(folder_name);
329                     fopen_s ( &fp, fname, "rb" ); //ファイルをバイナリ読み込み用に開く (ファイルの先頭にポインタを合わせる)
330                     read_header ( fp, 1, folder_name);  //ヘッダを読込み、画像の幅、高さ、1画素のビット数､全フレーム数を得る
331                     //SetDlgItemInt( hDlg, IDC_EDITallframe, FrameCount − 1, FALSE );   //フレーム番号の最大値は､全フレーム数 − 1
332                     SetDlgItemInt( hDlg, IDC_EDITstart, listctrl.start_no[i], FALSE );
333                     SetDlgItemInt( hDlg, IDC_EDIT1, listctrl.start_no[i], FALSE );    //フレーム番号の初期値を書き込む
334                     SetDlgItemInt( hDlg, IDC_EDITend, listctrl.end_no[i], FALSE );
335                     SetDlgItemInt( hDlg, IDC_EDIT_DETECT_LIMIT, listctrl.limit_no[i], FALSE ); //フレーム番号の初期値を書き込む
336                     SetDlgItemInt( hDlg, IDC_EDITthreshold, listctrl.threshold[i], FALSE);
337
338                     max_frame = FrameCount −1;
339                     ImageWidth2 = ImageWidth/2;
340                     ImageHeight2 = ImageHeight/2;//全体画像は実際の縦横 1/2倍のサイズにする。
341                     ImageSize = ImageWidth * ImageHeight;          //1フレームの全画素数を計算
342
343                     //16bit_ser 画像と 8bit_ser 画像の場合で分ける
344                     if( PixelDepthPerPlane == 16 ){
345                             value16 = (uint16 *)malloc( ImageSize * sizeof( uint16 ) );//フレーム 1枚分のメモリ確保
346                             value16_tmp = (uint16 *)malloc( ImageSize * sizeof( uint16 ) );//フレーム 1枚分のメモリ確保
347                     }
348                     else if ( PixelDepthPerPlane == 8 ) value8 = (uint8 *)malloc( ImageSize * sizeof( uint8 ) );
349                     else {
350                             MessageBox(NULL, "PixelDepthPerPlane が␣8␣or␣16␣でない", "ファイルがおかしい！", MB_OK);
351                             return TRUE;
352                     }
353
354                     //InitInstance0(NULL, SW_SHOWNORMAL); //フレーム全体表示用のウィンドウを表示する
355                     ser_read(fp);    //データ読み込み value16 に格納
356                     process0();    //表示用ビットマップを準備する//functions_image.cpp
357                     frame_display( 0 );  //フレームを表示する
358                     //MessageBox(NULL, listctrl.file_path[i], "debug", MB_OK );
359                     SendMessage (hDlg, WM_COMMAND, MAKEWPARAM(IDC_HIST, 0), 0);
360                     //DestroyWindow(g_img.hwin);
361                     //fclose(fp);
362             }//i < add_num
363             MessageBox(NULL,"終了:␣リストより検出処理を行いました。", "検出", MB_OK);
364             break;
365
366     case IDC_BUTTON_ADD:  //追加ボタン.検出リスト (右)にSER ファイルを追加する
367             hList = GetDlgItem(hDlg, IDC_LIST2);
368             if(add_num > 0){
369                     for(i = 0; i < add_num; i++){
370                             uiState = ListView_GetItemState(hList, i,LVIS_SELECTED | LVIS_FOCUSED);
371                             if ( uiState & LVIS_SELECTED ){        //選択されている行があったら
372                                     ListView_SetItemState(hList, i, 0, LVIS_SELECTED); //選択を解除する.
373                             }
374
375                     }
376             }
377             DlgDirSelectEx(hDlg, listctrl.list_fname, sizeof(listctrl.list_fname), IDC_LIST1);//選択したファイル名を list_fname にコピー
378             GetDlgItemText(hDlg, IDC_EDIT_PATH, listctrl.file_path[add_num], (int)sizeof(listctrl.file_path[add_num]));//フォルダパスを取得
                    //
379             strcat_s(listctrl.file_path[add_num], sizeof(listctrl.file_path[add_num]), slash);//"\"の追加
380             strcat_s(listctrl.file_path[add_num], sizeof(listctrl.file_path[add_num]), listctrl.list_fname);//ファイル名の追加、ファイルパス完成.
381             //MessageBox(NULL,listctrl.file_path[add_num], "listctrl.file_path の値",MB_OK);
382             if (error = fopen_s ( &fp, listctrl.file_path[add_num], "rb" )!=0){
383                     MessageBox(NULL,"ファイルが開けません (追加ボタン)", "debug",MB_OK); //ファイルをバイナリ読み込み用に開く (ファイルの先頭にポインタを合わせる)
384                     return TRUE;
385             }else{
386                     //MessageBox(NULL,"ファイルが開けました!", "debug",MB_OK);
387                     PathRemoveExtension(listctrl.list_fname);  //拡張子を削除
388                     read_header ( fp, 0, listctrl.list_fname);  //ヘッダを読込み、画像の幅、高さ、
                            1画素のビット数､全フレーム数を得る,第 3引数は検出用のフォルダを作るため.
389                     SetDlgItemInt( hDlg, IDC_EDITallframe, FrameCount − 1, FALSE );  //フレーム番号の最大値は、全フレーム数 − 1
390                     //SetDlgItemInt( hDlg, IDC_EDITend, 100, FALSE );//テスト用.実際はコメントアウトすること.
391                     SetDlgItemInt( hDlg, IDC_EDITend, FrameCount − 1, FALSE ); //テスト用にコメントアウト中
```

```
        .実際はここを使用.
392             SetDlgItemInt( hDlg, IDC_EDITstart, 0, FALSE);
393             //ファイル名
394             hList = GetDlgItem(hDlg, IDC_LIST2);
395             list_item.mask = LVIF_TEXT;
396             list_item.pszText = listctrl.list_fname;   // SER ファイル名
397             list_item.iItem = 0;        // 番号//おそらく行のこと
398             list_item.iSubItem = 0;        // サブアイテムの番号//おそらく列のこと
399             ListView_InsertItem(hList, &list_item);    //挿入する
400             //開始フレーム
401             listctrl.start_no[add_num] = GetDlgItemInt (hDlg, IDC_EDITstart, &check, TRUE);
402             wsprintf(listctrl.ch_start_no, "%d", listctrl.start_no[add_num]);
403             list_item.pszText = listctrl.ch_start_no;   //開始フレーム
404             list_item.iItem = 0;        // 番号
405             list_item.iSubItem = 1;        // サブアイテムの番号
406             ListView_SetItem(hList, &list_item);
407             //終了フレーム
408             listctrl.end_no[add_num] = GetDlgItemInt (hDlg, IDC_EDITend, &check, TRUE);
409             wsprintf(listctrl.ch_end_no, "%d",listctrl.end_no[add_num] );
410             list_item.pszText = listctrl.ch_end_no;
411             list_item.iItem = 0;        // 番号
412             list_item.iSubItem = 2;        // サブアイテムの番号
413             ListView_SetItem(hList, &list_item);
414             //上限
415             //listctrl.limit_no[add_num] = GetDlgItemInt (hDlg, IDC_EDIT_DETECT_LIMIT, &check, TRUE);
416             listctrl.limit_no[add_num] = listctrl.end_no[add_num] / 100 + 1;//検出上限フレーム数は、総フレーム数
                    /100 + 1とする
417             wsprintf(listctrl.ch_limit_no, "%d",listctrl.limit_no[add_num] );
418             list_item.pszText = listctrl.ch_limit_no;
419             list_item.iItem = 0;        // 番号
420             list_item.iSubItem = 3;        // サブアイテムの番号
421             ListView_SetItem(hList, &list_item);
422             //閾値
423             listctrl.threshold[add_num] = GetDlgItemInt (hDlg, IDC_EDITthreshold, &check, TRUE);
424             wsprintf(listctrl.ch_threshold, "%d",listctrl.threshold[add_num] );
425             list_item.pszText = listctrl.ch_threshold;
426             list_item.iItem = 0;        // 番号
427             list_item.iSubItem = 4;        // サブアイテムの番号
428             ListView_SetItem(hList, &list_item);
429             //総フレーム数
430             listctrl.allframe[add_num] = GetDlgItemInt(hDlg, IDC_EDITallframe, &check, TRUE);
431
432
433             max_frame = FrameCount -1;
434             //動画のサイズが大きいなら半分のサイズで表示
435             if(ImageWidth > 1500 && ImageHeight > 1000){//サイズが 1500 * 1000以上なら
436                     ImageWidth2 = ImageWidth/2;
437                     ImageHeight2 = ImageHeight/2;//全体画像は実際の縦横 1/2倍のサイズにする。
438             }else{
439                     ImageWidth2 = ImageWidth;
440                     ImageHeight2 = ImageHeight;
441             }
442             ImageSize = ImageWidth * ImageHeight;          //1フレームの全画素数を計算
443             if(add_num == 0) Cur_ImageSize = ImageSize;
444             //メモリの確保
445             if(add_num == 0 || Cur_ImageSize != ImageSize){//アイテムが無い or 違うサイズのアイテムが追加されたら
446                     free(value16);
447                     free(value16_tmp);
448                     if( PixelDepthPerPlane == 16 ){
449                             value16 = (uint16 *)malloc( ImageSize * sizeof( uint16 ) );//フレーム
                                    1枚分のメモリ確保
450                             value16_tmp = (uint16 *)malloc( ImageSize * sizeof( uint16 ) );//フレーム
                                    1枚分のメモリ確保
451                     }
452                     else if ( PixelDepthPerPlane == 8 ) value8 = (uint8 *)malloc( ImageSize * sizeof( uint8 )
                            );
453                     else {
454                             MessageBox(NULL, "PixelDepthPerPlane が␣8␣or␣16␣でない", "ファイルがおかしい！
                                    ", MB_OK);
455                             return TRUE;
456                     }
457             }
458             //ウィンドウの作成、サイズが異る場合サイズを変更
459             if(add_num == 0)InitInstance0(NULL, SW_SHOWNORMAL); //フレーム全体表示用のウィンドウを表示
                    する
460             else if(Cur_ImageSize != ImageSize){
461                     if(GetWindowRect(hDlg, &dlg_rect) == FALSE){//ダイアログボックスの位置を取得
462                             MessageBox(NULL, "ウィンドウの位置が特定できない","debug", MB_OK);
463                             return TRUE;
464                     }
465                     MoveWindow(g_img.hwin, dlg_rect.left - ImageWidth2, dlg_rect.top//ウィンドウのサイズを変更
466                             ,ImageWidth2 + GetSystemMetrics(SM_CXFIXEDFRAME) * 2 + 10,
                                    ImageHeight2 + GetSystemMetrics(SM_CYFIXEDFRAME) * 2 +
                                    GetSystemMetrics(SM_CYCAPTION) + 10 , TRUE);
467             }
468             //読み込んで表示
469             ser_read(fp);    //データ読み込み value16 に格納
470             if(add_num == 0 || Cur_ImageSize != ImageSize) {//ウィンドウが無いまたは異なるサイズの時
471                     process0();    //表示用ビットマップを準備する//functions_image.cpp
```

```
472                         Cur_ImageSize = ImageSize;
473                 }
474                 SetDlgItemInt( hDlg, IDC_EDIT1, c_frame, FALSE ); //フレーム番号の初期値を書き込む
475                 frame_display( 0 );  //フレームを表示する
476                 SetWindowText(g_img.hwin,listctrl.file_path[add_num]);//ウィンドウの名前を変更.
477                 //fclose(fp);
478                 if(add_num < LIST_ITEM_MAX) add_num++;//今のところファイルは 20個まで.
479                 else MessageBox(NULL, "ファイルを開きすぎです。","debug",MB_OK);
480                 //char test[100];
481                 //wsprintf(test, "add_num = %d",add_num);
482                 //MessageBox(NULL,test,"debug", MB_OK);
483         }
484         break;
485
486 case IDC_BUTTON_REMOVE:  //削除ボタン.検出リストからSER ファイルを削除する
487         hList = GetDlgItem(hDlg, IDC_LIST2);
488         //char ctes5[200];
489
490         if(add_num > 0)//リストに何もないなら削除しない.
491         {
492                 for(i = 0; i < add_num; i++){
493                         uiState = ListView_GetItemState(hList, i,LVIS_SELECTED | LVIS_FOCUSED);
494                         if ( uiState & LVIS_SELECTED ){//行が選択されているなら
495                                 /*//選択されている行NO を確認.
496                                 //wsprintf(ctes5,"i=%d, add_num=%d",i, add_num);
497                                 //MessageBox(NULL, ctes5, "選択されている行は", MB_OK);
498                                 */
499                                 ListView_DeleteItem(hList, i);    //選択されている行を削除.
500                                 //リスト 2の情報を更新
501                                 for(i;i > 0; i--){//i=0すなわち
502                                         for(int j=0; j < MAX_PATH; j++){
503                                                 listctrl.file_path[add_num - i-1][j] = listctrl.file_path[add_num - i
                                                        ][j] ;//パ
                                                        ス
504                                         }
505                                         //パスと開始 NO など以外は変更しなくても OK
506                                         //listctrl.list_fname[20];//ファイル名
507                                         //listctrl.ch_start_no[10];//開始フレーム
508                                         //listctrl.ch_end_no[10];//終了フレーム
509                                         //listctrl.ch_limit_no[10];
510                                         listctrl.start_no[add_num - i-1] = listctrl.start_no[add_num - i];
511                                         listctrl.end_no[add_num - i-1] =listctrl.end_no[add_num - i] ;
512                                         listctrl.limit_no[add_num -i-1] = listctrl.limit_no[add_num - i];
513                                         listctrl.threshold[add_num - i-1] = listctrl.threshold[add_num - i];
514                                         listctrl.allframe[add_num - i-1] = listctrl.allframe[add_num - i];
515                                         }
516                                 if(add_num == 1) DestroyWindow(g_img.hwin); //アイテムが無くなる時は全体画面も削除.
517                                 add_num--;//カウンターを更新
518                                 break;//for を抜ける
519                         }
520                 }
521         }
522         break;
523 case IDC_BUTTON_ITEM_EDIT: //編集ボタン.検出リスト内の値を編集する//http://oshiete.goo.ne.jp/qa/3221910.html 参照
524         //char ctes[40];
525         hList = GetDlgItem(hDlg, IDC_LIST2);
526         for(i = 0; i < add_num; i++){
527                 uiState = ListView_GetItemState(hList, i,LVIS_SELECTED | LVIS_FOCUSED);
528                 if ( uiState & LVIS_SELECTED ){// 選択されている
529                         //wsprintf(ctes, "%d 行目が選択されている",i);
530                         //MessageBox(NULL, ctes,"debug",MB_OK);
531                         //ListView_DeleteItem(hList, i);//アイテムを削除する
532                         list_item.mask = LVIF_TEXT;
533                         //list_item.pszText = TEXT("編集中");//listctrl.list_fname;   // SER ファイル名
534                         //list_item.iItem = i;         // 番号//おそらく行のこと
535                         //list_item.iSubItem = 0;         // サブアイテムの番号//おそらく列のこと
536                         //ListView_InsertItem(hList, &list_item);    //挿入する
537
538                         listctrl.start_no[ add_num - i -1] = GetDlgItemInt (hDlg, IDC_EDITstart, &check, TRUE
                                );//add_num - i - 1 にするのは配列と行の番号が逆であるため.
539                         wsprintf(listctrl.ch_start_no, "%d", listctrl.start_no[ add_num - i - 1]);
540                         list_item.pszText = listctrl.ch_start_no;   //開始フレーム
541                         list_item.iItem = i;         // 番号
542                         list_item.iSubItem = 1;         // サブアイテムの番号
543                         ListView_SetItem(hList, &list_item);
544
545                         listctrl.end_no[ add_num - i - 1] = GetDlgItemInt (hDlg, IDC_EDITend, &check, TRUE);
546                         wsprintf(listctrl.ch_end_no, "%d",listctrl.end_no[ add_num - i -1] );
547                         list_item.pszText = listctrl.ch_end_no;    //開始フレーム
548                         list_item.iItem = i;         // 番号
549                         list_item.iSubItem = 2;         // サブアイテムの番号
550                         ListView_SetItem(hList, &list_item);
551
552                         listctrl.limit_no[ add_num - i - 1] = GetDlgItemInt (hDlg, IDC_EDIT_DETECT_LIMIT, &
                                check, TRUE);
553                         wsprintf(listctrl.ch_limit_no, "%d",listctrl.limit_no[ add_num - i -1] );
554                         list_item.pszText = listctrl.ch_limit_no;    //開始フレーム
555                         list_item.iItem = i;         // 番号
```

```
556                             list_item.iSubItem = 3;         // サブアイテムの番号
557                             ListView_SetItem(hList, &list_item);
558
559                             listctrl.threshold[ add_num − i − 1] = GetDlgItemInt (hDlg, IDC_EDITthreshold, &check,
                                    TRUE);
560                             wsprintf(listctrl.ch_threshold, "%d",listctrl.threshold[ add_num − i −1] );
561                             list_item.pszText = listctrl.ch_threshold;    //開始フレーム
562                             list_item.iItem = i;          // 番号
563                             list_item.iSubItem = 4;         // サブアイテムの番号
564                             ListView_SetItem(hList, &list_item);
565                             //総フレーム数は変わらない
566
567                     }
568                     else if( uiState & LVIS_FOCUSED ){// フォーカスがある
569                             //MessageBox(NULL, "0がフォーカスされている","debug",MB_OK);
570                     }
571                     else{// 選択されていない
572                             //MessageBox(NULL, "選択されていない。","debug",MB_OK);}
573                     }
574             }
575             break;
576
577
578     case IDC_BUTTON1:   //ファイルを開く処理.//残骸
579             if( fname = file_open( ) ){
580                             //char ch[100];
581                             //wsprintf(ch, "%s", fname);
582                             //MessageBox(NULL, ch, "debug",MB_OK);//フレームを表示する
583                             //threshold = GetDlgItemInt (hDlg, IDC_EDITthreshold, &check, TRUE);
584                             //MessageBox(NULL, "配列を初期化します", "MessageBox", MB_OK);
585                             wsprintf(ch, TEXT("%s"), fname);
586                             folder_name = PathFindFileName(ch);
587                             PathRemoveExtension(folder_name);
588                             //Tech(folder_name);
589                             //Tech(fname);
590
591                             //wsprintf(detection_path, "Detection_%s\\SER_Detection_Result.txt", folder_name);//検出ログ
                                    用パス
592                             //Tech(folder_name);
593
594                             //ファイルのパス名を得る
595                             fopen_s ( &fp, fname, "rb" ); //ファイルをバイナリ読み込み用に開く
                                    (ファイルの先頭にポインタを合わせる)
596                             //fp = read_header ( fp, 0, folder_name );  //ヘッダを読込み、画像の幅、高さ、1画素のビット数、全フレ
                                    ーム数を得る
597                             read_header ( fp, 1, folder_name);  //ヘッダを読込み、画像の幅、高さ、
                                    1画素のビット数、全フレーム数を得る
598
599                             SetDlgItemInt( hDlg, IDC_EDITallframe, FrameCount − 1, FALSE );  //フレーム番号の最大値
                                    は、全フレーム数 − 1
600
601                             //SetDlgItemInt( hDlg, IDC_EDITend, 100, FALSE );//テスト用.実際はコメントアウトすること.
602                             SetDlgItemInt( hDlg, IDC_EDITend, FrameCount − 1, FALSE ); //テスト用にコメントアウト中
                                    .実際はここを使用.
603
604                             max_frame = FrameCount −1;
605                             ImageWidth2 = ImageWidth/2;
606                             ImageHeight2 = ImageHeight/2;//全体画像は実際の縦横 1/2倍のサイズにする。
607                             ImageSize = ImageWidth * ImageHeight;          //1フレームの全画素数を計算
608
609                             //16bit_ser 画像と 8bit_ser 画像の場合で分ける
610                             if( PixelDepthPerPlane == 16 ){
611                                     value16 = (uint16 *)malloc( ImageSize * sizeof( uint16 ) );//フレーム
                                            1枚分のメモリ確保
612                                     value16_tmp = (uint16 *)malloc( ImageSize * sizeof( uint16 ) );//フレーム
                                            1枚分のメモリ確保
613                             }
614                             else if ( PixelDepthPerPlane == 8 ) value8 = (uint8 *)malloc( ImageSize * sizeof( uint8 )
                                    );
615                             else {
616                                     MessageBox(NULL, "PixelDepthPerPlane が␣8␣or␣16␣でない", "ファイルがおかしい！
                                            ", MB_OK);
617                                     return TRUE;
618                             }
619
620                             //max_std = ( double * )malloc( ImageSize * sizeof( double ) );
621
622                             InitInstance0(NULL, SW_SHOWNORMAL); //フレーム全体表示用のウィンドウを表示する
623                             ser_read(fp);    //データ読み込み value16 に格納
624                             process0();    //表示用ビットマップを準備する//functions_image.cpp
625                             SetDlgItemInt( hDlg, IDC_EDIT1, c_frame, FALSE ); //フレーム番号の初期値を書き込む
626                             frame_display( 0 );  //フレームを表示する
627
628             }
629             return TRUE;
630
631     case IDC_BUTTON_FRAME:  //検出リスト上で選択された動画の指定されたフレームの表示.
632             hList = GetDlgItem(hDlg, IDC_LIST2);
633             //選択されているアイテムを特定し、ファイルオープン.
634             for(i = 0; i < add_num; i++){//上から 0, 1, 2,…//DLIST は下から 0, 1,…
```

```
635                     uiState = ListView_GetItemState(hList, i,LVIS_SELECTED | LVIS_FOCUSED);
636                     if ( uiState & LVIS_SELECTED ){// 選択されている行を特定
637                             SetWindowText(g_img.hwin,listctrl.file_path[add_num − i −1]);//ウィンドウの名前を変更.
638                             if (error = fopen_s ( &fp, listctrl.file_path[add_num − i −1], "rb" )!=0){//ファイルを開く
639                                     MessageBox(NULL,"ファイルが開けません (指定されたフレームを表示)", "debug",MB_OK);
640                                     return TRUE;
641                             }
642                             else {
643                                     read_header ( fp, 0, listctrl.file_path[add_num − i −1]);
644                                     if(ImageWidth > 1500 && ImageHeight > 1000){//サイズが 1500 * 1000以上なら
645                                             ImageWidth2 = ImageWidth/2;
646                                             ImageHeight2 = ImageHeight/2;//全体画像は実際の縦横 1/2倍のサイズにする。
647                                     }else{
648                                             ImageWidth2 = ImageWidth;
649                                             ImageHeight2 = ImageHeight;
650                                     }
651                                     ImageSize = ImageWidth * ImageHeight;          //1フレームの全画素数を計算
652                                             if(Cur_ImageSize != ImageSize) {//ウィンドウのサイズが異なるなら
653                                             //MessageBox(NULL, "サイズが異なります","debug", MB_OK);
654                                             //メモリ確保
655                                             free(value16);
656                                             free(value16_tmp);
657                                             if( PixelDepthPerPlane == 16 ){
658                                                     value16 = (uint16 *)malloc( ImageSize * sizeof( uint16 ) );//フレーム 1枚分のメモリ確保
659                                                     value16_tmp = (uint16 *)malloc( ImageSize * sizeof( uint16 ) );//フレーム
                                                             1枚分のメモリ確保
660                                             }
661                                             else if ( PixelDepthPerPlane == 8 ) value8 = (uint8 *)malloc( ImageSize
                                                     * sizeof( uint8 ) );
662                                             else {
663                                                     MessageBox(NULL, "PixelDepthPerPlane が␣8␣or␣16␣でない", "ファイルがおかしい！", MB_OK);
664                                                     return TRUE;
665                                             }
666                                             //ウィンドウサイズの変更
667                                             if(GetWindowRect(hDlg, &dlg_rect) == FALSE){//ダイアログボックスの位置を取得
668                                                     MessageBox(NULL, "ウィンドウの位置が特定できない","debug", MB_OK);
669                                                     return TRUE;
670                                             }
671                                             MoveWindow(g_img.hwin, dlg_rect.left − ImageWidth2 , dlg_rect.top//ウィンドウのサイズを変更
672                                                     ,ImageWidth2 + GetSystemMetrics(SM_CXFIXEDFRAME) * 2 +
                                                             10, ImageHeight2 + GetSystemMetrics(
                                                             SM_CYFIXEDFRAME) * 2 + GetSystemMetrics(
                                                             SM_CYCAPTION) + 10 , TRUE);
673                                             process0();      //表示用ビットマップを準備する//functions_image.cpp
674                                             Cur_ImageSize = ImageSize;  //現在の ImageSize を更新
675                                     }
676                             }
677                     }
678             }
679             //カレントフレーム番号が範囲外なら範囲内に編集
680             max_frame = GetDlgItemInt(hDlg, IDC_EDITallframe, &check, TRUE);
681             c_frame = GetDlgItemInt( hDlg, IDC_EDIT1, &check, TRUE ); //フレーム番号を得る
682             if ( c_frame > max_frame ) {            //フレーム番号がその最大値を越えていた場合、最大値に設定.
683                     c_frame = max_frame;
684                     SetDlgItemInt( hDlg, IDC_EDIT1, c_frame, FALSE );
685             }
686             if ( c_frame < 0 ) {                //フレーム番号が負の場合、0にする.
687                     c_frame = 0;
688                     SetDlgItemInt( hDlg, IDC_EDIT1, c_frame, FALSE );
689             }
690             //読み込んで表示
691             _fseeki64 ( fp, (__int64)headersize + (__int64)c_frame * (__int64)ImageSize * 2, SEEK_SET );
692             ser_read(fp);
693             frame_display(0);     //フレームを表示する
694
695             return TRUE;
696
697     case IDC_BUTTON_PREV:  //前フレームの表示
698             if( 0 < c_frame ){
699                     c_frame−−;
700                     SetDlgItemInt( hDlg, IDC_EDIT1, c_frame, FALSE );  //フレーム番号を書き換える
701
702                     _fseeki64 ( fp, −(__int64)ImageSize*2*2, SEEK_CUR );     //ImageSize*2(=画素一つ 16bit) * 2枚 [byte ] 分へ
703                     //char ch[100];
704                     //wsprintf(ch, "%I64d", _ftelli64(fp));
705                     //wsprintf(ch, "c_frame = %d, max_frame = %d", c_frame, max_frame);
706                     //MessageBox(NULL,ch,"",MB_OK);
707                     ser_read(fp);
708                     frame_display(0);     //フレームを表示する
709             }
710             return TRUE;
711
712     case IDC_BUTTON_NEXT:  //次フレームの表示
```

```
713             if( 0 <= c_frame && c_frame < max_frame ){
714                     c_frame++;
715                     SetDlgItemInt( hDlg, IDC_EDIT1, c_frame, FALSE );  //フレーム番号を書き換える
716                     //char ch[100];
717                     //wsprintf(ch, "%I64d",_ftelli64(fp));
718                     //wsprintf(ch, "c_frame = %d, max_frame = %d", c_frame, max_frame);
719                     //MessageBox(NULL,ch,"",MB_OK);
720                     ser_read(fp);
721                     frame_display(0);     //フレームを表示する
722             }
723             return TRUE;
724
725     case IDC_HIST:    //検出処理
726             wsprintf(ch, TEXT("%s"), fname);
727             folder_name = PathFindFileName(ch);
728             PathRemoveExtension(folder_name);
729
730             wsprintf(detect_dir, "検出_%s", folder_name);
731             _mkdir(detect_dir);//予備
732             wsprintf(detect_result_path, "%s\\SER_Detection_Result.txt", detect_dir);
733             //ch(detect_dir);
734             //ch(detect_result_path);
735             //Tech(fname);
736             //read_header(fp, 1, folder_name);
737             //read_header(fp, 1);//1はヘッダ情報を記述する.
738
739             allframe = GetDlgItemInt (hDlg, IDC_EDITallframe, &check, TRUE); //総フレーム数を得る
740             end = GetDlgItemInt (hDlg, IDC_EDITend, &check, TRUE); //終了フレーム No.を得る
741             hani = GetDlgItemInt (hDlg, IDC_EDIThani, &check, TRUE); //処理するフレーム数を得る
742             threshold = GetDlgItemInt (hDlg, IDC_EDITthreshold, &check, TRUE);
743             hidari = GetDlgItemInt (hDlg, IDC_EDIThidari, &check, TRUE);
744             migi = GetDlgItemInt (hDlg, IDC_EDITmigi, &check, TRUE);
745             ue = GetDlgItemInt (hDlg, IDC_EDITue, &check, TRUE);
746             shita = GetDlgItemInt (hDlg, IDC_EDITshita, &check, TRUE);
747             c_frame = GetDlgItemInt( hDlg, IDC_EDIT1, &check, TRUE ); //フレーム番号を得る
748             detect_limit = GetDlgItemInt( hDlg, IDC_EDIT_DETECT_LIMIT, &check, TRUE );
749             int square_limit;//四角を描く数に上限をつける。
750
751             //calloc メモリ領域を確保して 0で初期化する。(個数, 1 つあたりのバイト数)
752             x2heikin = ( double * )calloc( ImageSize, sizeof( double ) ); //平均計算のために輝度値の合計を入れる
753             ave   = ( double * )calloc( ImageSize, sizeof( double ) );
754             sigma  = ( double * )calloc( ImageSize, sizeof( double ) ); //平均値を入れる
755
756             /*if (Button_GetCheck(hCheck1) == BST_CHECKED){
757                     ue = ue * (-1);
758             }
759             if (Button_GetCheck(hCheck2) == BST_CHECKED){
760                     shita = shita * (-1);
761             }*/
762             number = 0;
763             count = 0;
764             repeat = (end - c_frame) / hani;
765             if ((end - c_frame + 1) % hani != 0) { //処理間隔で余りが出るなら
766                     repeat ++;       //処理間隔を変更して最後までやる
767                     last_hani = (end - c_frame) % hani; //最後の範囲は余り
768             } else{
769                     last_hani = hani;
770             }
771             FILE *file;
772
773             if(error = fopen_s(&file, detect_result_path, "a") !=0 ){
774                     MessageBox(NULL, "ファイルが開けません (検出ボタン)", "debug",MB_OK);
775             }
776
777     //検出処理開始.
778             for (int m = 0; number < detect_limit && number < allframe && threshold > 0 ; m++){//検出上限フレーム
                        数に達していない, 検出数が総フレーム数未満, threshold が 1 以上なら
779
780
781                     if(m > 0){
782                             hani = GetDlgItemInt (hDlg, IDC_EDIThani, &check, TRUE); //処理するフレーム数を得る
783                             threshold -= 5;//スレッショルドを 1下げる。
784                             //c_frame -= repeat*hani;//開始フレームを元に戻す.
785                             if(hani != last_hani) c_frame -= (repeat-1)*hani + last_hani;//余りが出る場合は更にその余り分
                                    だけ戻す.
786                             else c_frame -= repeat*hani;
787                             SetDlgItemInt( hDlg, IDC_EDITthreshold, threshold, FALSE );
788                             SetDlgItemInt( hDlg, IDC_EDIT1, c_frame, FALSE );
789
790                             //MessageBox(NULL,"検出上限フレーム数に達しませんでした。閾値を下げます。\n","debug",MB_OK);
791
792                     }
793
794                     _fseeki64 ( fp, (__int64)headersize + (__int64)c_frame * (__int64)ImageSize * 2, SEEK_SET );     //フ
                            レームデータの先頭にポインタを合わせる
795                     ser_read(fp);//データを読み込む
796                     fprintf(file, "Detection Information----------\n");
797                     //fprintf(file, "ファイル名: %s\n", folder_name);
798                     fprintf(file, "開始フレーム No. %d 終了フレームNo. %d 処理間隔:%d フレーム\n 閾値: %d σ\ 上=%d%% 下=%d%%
```

```
                    ␣右=%d%%␣␣左=%d%%␣\n",c_frame, end, hani, threshold,ue,shita,migi,hidari);
799                 if(Button_GetCheck(hRadio_median_on) == BST_CHECKED){//
800                     fprintf(file, "メディアンフィルタ:␣ON\n");
801                 }else if(Button_GetCheck(hRadio_median_off) == BST_CHECKED){//
802                     fprintf(file, "メディアンフィルタ:␣OFF\n");
803                 }
804                 fprintf(file, "検出箇所の情報を保存します。※ 65534行以上は保存出来ません。\n");
805                 fprintf(file,"\n%-10s%-10s%-10s%-10s%-10s%-10s%-10s\n", "No.","Frame","横座標", "縦座標", "画素値", "平均値
                        ", "標準偏差");
806
807 //処理間隔毎に繰り返す.
808                 for(l=0; l<repeat; l++){
809                     if(l == repeat−1) hani = last_hani;
810                     //char ch[100];
811                     //wsprintf(ch, "c_frame = %d", c_frame);
812                     //MessageBox(NULL, ch, "debug",MB_OK);//フレームを表示する
813                     //threshold = GetDlgItemInt (hDlg, IDC_EDITthreshold, &check, TRUE);
814                     //MessageBox(NULL, "配列を初期化します", "MessageBox", MB_OK);
815                     filesave = 0;   //この繰り返しの中でで検出されたかどうかを判断する変数
816                     for (i = 0; i < ImageHeight; i++){
817                         for (j = 0; j < ImageWidth; j++){
818                             ave[ImageWidth*i + j] = 0.0; //平均の配列初期化
819                             sigma[ImageWidth*i + j] = 0.0; //平均の配列初期化
820                             x2heikin[ImageWidth*i + j] = 0.0; //x^2の平均の配列初期化
821                         }
822                     }
823 //MessageBox(NULL, "標準偏差を計算します", "MessageBox", MB_OK);
824                     for (n = 0; n < hani; n++){//n は繰り返し回数
825                         //ser_read(fp);//データを読み込む
826                         for (i = 0; i < ImageHeight; i++){
827                             for (j = 0; j < ImageWidth; j++){
828                                 if(i<10 || i>ImageHeight−10 || j<10 || j>ImageWidth−10){//画像
                                        の端でエラーが出るのを避けている。
829                                     value16[ImageWidth*i + j] = 0;
830                                 }
831                                 else{
832                                     ave[ImageWidth*i + j] = ave[ImageWidth*i + j] + (
                                            double)value16[ImageWidth*i +j];//ave に合計を格納
833                                     x2heikin[ImageWidth*i + j] += ((double)value16[
                                            ImageWidth*i +j]) * ((double)value16[ImageWidth
                                            *i +j]);//x2heikin に 2 乗和を格納
834                                 }
835                             }
836                         }
837                         SendMessage (hDlg, WM_COMMAND, MAKEWPARAM(IDC_BUTTON_NEXT,
                                0), 0); //次のフレームへ
                                進む
838                     }
839
840                     for (i = 0; i < ImageHeight; i++){
841                         for (j = 0; j < ImageWidth; j++){
842                             ave[ImageWidth*i + j] = ave[ImageWidth*i + j] / (double) hani;//平均値を
                                    計算
843                             x2heikin[ImageWidth*i + j] = x2heikin[ImageWidth*i + j] / (double)
                                    hani;//2乗和の平均値
844                             sigma[ImageWidth*i + j] = sqrt((double)x2heikin[ImageWidth*i + j] − ((
                                    ave[ImageWidth*i + j]) * (ave[ImageWidth*i + j])));//標準偏差を
                                    計算
845                         }
846                     }
847 //MessageBox(NULL,"標準偏差計算完了","debug",MB_OK);
848 //処理間隔のはじめのフレームに戻る
849                     c_frame −= hani;
850                     SetDlgItemInt( hDlg, IDC_EDIT1, c_frame, FALSE );  //フレーム番号を書き換える
851                     //SendMessage (hDlg, WM_COMMAND, MAKEWPARAM(IDC_BUTTON_FRAME, 0), 0);
                            //指定されたフレームへ移動
852                     _fseeki64 ( fp, (__int64)headersize + (__int64)c_frame * (__int64)ImageSize * 2, SEEK_SET
                            );      //フレームデータの先頭にポインタを合わ
                            せる
853                     ser_read(fp);
854 //検出 フレーム, i, j の値を配列に格納
855 for (n = 0; n < hani; n++){
856         detect = 0;
857         square_limit=0;
858 for (i = 10; i < ImageHeight−10; i++) {
859 for (j = 10; j < ImageWidth−10; j++){
860 if (value16[ImageWidth*i +j]*10 > ((int)ave[ImageWidth*i + j]*10 +(int)sigma[ImageWidth*i + j] * threshold) ){
        //標準偏差の閾値倍大きけ
        れば
861 if(  //(value16[ImageWidth*i +j−1] == 0 ||//??
862 (((double)value16[ImageWidth*i + j−1] − ave[ImageWidth*i + j−1] )
863 /((double)value16[ImageWidth*i+j] − ave[ImageWidth*i + j]) * 100 >= hidari)
864
865 && //(value16[ImageWidth*i +j+1] == 0 ||
866 (((double)value16[ImageWidth*i +j+1] − ave[ImageWidth*i +j+1] )
867 /((double)value16[ImageWidth*i +j]− ave[ImageWidth*i + j]) * 100 >= migi)
868
869 && //(value16[ImageWidth*(i−1) +j]== 0 ||
870 (((double)value16[ImageWidth*(i−1) +j] − ave[ImageWidth*(i−1) +j]) //インターレースは考慮していない。
871 /((double)value16[ImageWidth*i +j] − ave[ImageWidth*i + j]) * 100 >= ue)
```

```
872
873   && //(value16[ImageWidth*(i+1) +j] == 0 ||
874   (((double)value16[ImageWidth*(i+1) +j] - ave[ImageWidth*(i+1) +j] )
875   /((double)value16[ImageWidth*i +j] - ave[ImageWidth*i + j]) * 100 >= shita)
876   )
877   {
878   square_limit++;
879   count += 1;//検出画素数
880
881   //ログ書き込み
882   if(count<SHRT_MAX*2){
883   fprintf (file, "%-10d%-10d%-10d%-10d%-10d%-10d%-10d\n", count, c_frame, j, i, (int)value16[ImageWidth*i +j] , (int)ave[
          ImageWidth*i + j], (int)sigma[ImageWidth*i + j]);
884
885   }
886   filesave = 1;
887
888   //検出箇所に赤い□を描く.ただし、square_limit より多くある場合は描画しない.
889   if(square_limit < SQUARE_LIMIT){
890   int ij = i*ImageWidth + j;
891   for (square = (ij-ImageWidth*5-5); square <= (ij-ImageWidth*5+5); square++){ //下辺
892           g_img.lpBmpData[square*3] = 0;
893           g_img.lpBmpData[square*3+1] = 0;
894           g_img.lpBmpData[square*3+2] = 200;
895   }
896   g_img.lpBmpData[(ij-ImageWidth*4-5)*3] = 0; g_img.lpBmpData[(ij-ImageWidth*4+5)*3] = 0;
897   g_img.lpBmpData[(ij-ImageWidth*3-5)*3] = 0; g_img.lpBmpData[(ij-ImageWidth*3+5)*3] = 0;
898   g_img.lpBmpData[(ij-ImageWidth*2-5)*3] = 0; g_img.lpBmpData[(ij-ImageWidth*2+5)*3] = 0;
899   g_img.lpBmpData[(ij-ImageWidth-5)*3] = 0; g_img.lpBmpData[(ij-ImageWidth+5)*3] = 0;
900   g_img.lpBmpData[(ij-5)*3] = 0; g_img.lpBmpData[(ij+5)*3] = 0;
901   g_img.lpBmpData[(ij+ImageWidth-5)*3] = 0; g_img.lpBmpData[(ij+ImageWidth+5)*3] = 0;
902   g_img.lpBmpData[(ij+ImageWidth*2-5)*3] = 0; g_img.lpBmpData[(ij+ImageWidth*2+5)*3] = 0;
903   g_img.lpBmpData[(ij+ImageWidth*3-5)*3] = 0; g_img.lpBmpData[(ij+ImageWidth*3+5)*3] = 0;
904   g_img.lpBmpData[(ij+ImageWidth*4-5)*3] = 0; g_img.lpBmpData[(ij+ImageWidth*4+5)*3] = 0;
905   g_img.lpBmpData[(ij-ImageWidth*4-5)*3+1] = 0; g_img.lpBmpData[(ij-ImageWidth*4+5)*3+1] = 0;
906   g_img.lpBmpData[(ij-ImageWidth*3-5)*3+1] = 0; g_img.lpBmpData[(ij-ImageWidth*3+5)*3+1] = 0;
907   g_img.lpBmpData[(ij-ImageWidth*2-5)*3+1] = 0; g_img.lpBmpData[(ij-ImageWidth*2+5)*3+1] = 0;
908   g_img.lpBmpData[(ij-ImageWidth-5)*3+1] = 0; g_img.lpBmpData[(ij-ImageWidth+5)*3+1] = 0;
909   g_img.lpBmpData[(ij-5)*3+1] = 0; g_img.lpBmpData[(ij+5)*3+1] = 0;
910   g_img.lpBmpData[(ij+ImageWidth-5)*3+1] = 0; g_img.lpBmpData[(ij+ImageWidth+5)*3+1] = 0;
911   g_img.lpBmpData[(ij+ImageWidth*2-5)*3+1] = 0; g_img.lpBmpData[(ij+ImageWidth*2+5)*3+1] = 0;
912   g_img.lpBmpData[(ij+ImageWidth*3-5)*3+1] = 0; g_img.lpBmpData[(ij+ImageWidth*3+5)*3+1] = 0;
913   g_img.lpBmpData[(ij+ImageWidth*4-5)*3+1] = 0; g_img.lpBmpData[(ij+ImageWidth*4+5)*3+1] = 0;
914   g_img.lpBmpData[(ij-ImageWidth*4-5)*3+2] = 200; g_img.lpBmpData[(ij-ImageWidth*4+5)*3+2] = 200;
915   g_img.lpBmpData[(ij-ImageWidth*3-5)*3+2] = 200; g_img.lpBmpData[(ij-ImageWidth*3+5)*3+2] = 200;
916   g_img.lpBmpData[(ij-ImageWidth*2-5)*3+2] = 200; g_img.lpBmpData[(ij-ImageWidth*2+5)*3+2] = 200;
917   g_img.lpBmpData[(ij-ImageWidth-5)*3+2] = 200; g_img.lpBmpData[(ij-ImageWidth+5)*3+2] = 200;
918   g_img.lpBmpData[(ij-5)*3+2] = 200; g_img.lpBmpData[(ij+5)*3+2] = 200;
919   g_img.lpBmpData[(ij+ImageWidth-5)*3+2] = 200; g_img.lpBmpData[(ij+ImageWidth+5)*3+2] = 200;
920   g_img.lpBmpData[(ij+ImageWidth*2-5)*3+2] = 200; g_img.lpBmpData[(ij+ImageWidth*2+5)*3+2] = 200;
921   g_img.lpBmpData[(ij+ImageWidth*3-5)*3+2] = 200; g_img.lpBmpData[(ij+ImageWidth*3+5)*3+2] = 200;
922   g_img.lpBmpData[(ij+ImageWidth*4-5)*3+2] = 200; g_img.lpBmpData[(ij+ImageWidth*4+5)*3+2] = 200;
923   for (square = (ij+ImageWidth*5-5); square <= (ij+ImageWidth*5+5); square++){//上辺
924           g_img.lpBmpData[square*3] = 0;
925           g_img.lpBmpData[square*3+1] = 0;
926           g_img.lpBmpData[square*3+2] = 200;
927   }
928   }//square_limit
929   detect = 1;
930   }//周囲の画素の条件
931   }//標準偏差の条件
932   }//j の終わり
933   }//i の終わり
934   //検出箇所があったフレームを, 順番に名前を付けてビットマップファイルで保存
935
936           if (detect == 1){
937                   FILE *fp1;
938                   if (allframe >= 100000){
939                   // sprintf(fname, "%08d_%04d_%06d.bmp", date, number, c_frame);
940                           wsprintf(fname, "%s\\%04d_s%d_%06d.bmp",detect_dir, number, threshold, c_frame); //検出ナンバー, 閾値 , フレームナンバー
941                   }
942                   else{
943                           //sprintf(fname, "%08d_%04d_%05d.bmp", date, number, c_frame);
944                           wsprintf(fname, "%s\\%04d_s%d_%05d.bmp",detect_dir, number, threshold, c_frame);
945                   }
946   //bmp 書き出し
947                   fopen_s(&fp1, fname, "wb" );
948                   //ファイルヘッダーの準備
949                   BITMAPFILEHEADER bfh;
950                   bfh.bfType = 'M'*256+'B';
951                   bfh.bfReserved1 = 0;
952                   bfh.bfReserved2 = 0;
953                   bfh.bfOffBits = sizeof(BITMAPFILEHEADER)+ sizeof(BITMAPINFOHEADER);
954                   bfh.bfSize = bfh.bfOffBits + g_img.bmp_ih.biSizeImage;
955                   fwrite( &bfh, sizeof(BITMAPFILEHEADER), 1, fp1 );   //ファイルヘッダー
956                   fwrite( &( g_img.bmp_ih ), sizeof(BITMAPINFOHEADER), 1, fp1 ); //インフォヘッダー
957                   fwrite( g_img.lpBmpData , 1, g_img.bmp_ih.biSizeImage, fp1 ); //データを書き出す
```

```
958                     fclose(fp1);
959
960                     if(number >= detect_limit-1) {//検出上限数に到達したら終了.
961                             fprintf(file, "検出上限フレーム数に達しました。検出フレーム数は%d です。\n", number+1);
962                             fprintf(file, "------------------------------------------------------------\n\n");
963                             fclose(file);
964                             //SetDlgItemInt( hDlg, IDC_EDITjikkou, repeat * hani, FALSE ); //実行フレーム数を
                                    EditBox に書き込む.
965                             SetDlgItemInt( hDlg, IDC_EDIT1, c_frame, FALSE ); //現在のフレーム数を EditBox に書き込む.
966                             number = 0;//検出数を元に戻す.
967                             if(add_num == 0)MessageBox(NULL, "終了:検出上限数に達しました。", "MessageBox", MB_OK);
968                             //最終的な閾値とフレーム数をフォルダ名に追加する.
969                             wsprintf(detect_dir_c_frame, "%s_s%d_%d", detect_dir,threshold, c_frame);
970                             rename(detect_dir,detect_dir_c_frame);
971                             /*
972                             if(rename(detect_dir,detect_dir_c_frame) == 0){
973                                     MessageBox(NULL, "フォルダ名変更した。","debug", MB_OK);
974                             }else
975                                     MessageBox(NULL, "フォルダ名変更出来ない。","debug", MB_OK);
976                             */
977
978                             return TRUE;
979                     }
980                     number += 1;
981             }
982
983             SendMessage (hDlg, WM_COMMAND, MAKEWPARAM(IDC_BUTTON_NEXT, 0), 0); //次のフレームへ進む
984     }//範囲 hani の終わり
985                     } // 繰り返し"l"の終わり//範囲分の計算を何回行うか
986                     if(count == 0)fprintf(file, "検出条件に該当するフレームはありませんでした。\n");
987                     else fprintf(file, "検出フレーム数は%d です。\n", number);
988                     fprintf(file, "------------------------------------------------------------\n\n");
989     //自動更新 OFF にチェックが入っている場合は終了.
990                     if (Button_GetCheck(hRadio_auto_update_off) == BST_CHECKED) {
991                             fclose(file);
992                             _fseeki64 ( fp, (__int64)headersize + (__int64)c_frame * (__int64)ImageSize * 2, SEEK_SET
                                    );
993                             ser_read(fp);
994
995                             //SetDlgItemInt( hDlg, IDC_EDITjikkou, repeat * hani, FALSE ); //実行フレーム数を
                                    EditBox に書き込む
996                             SetDlgItemInt( hDlg, IDC_EDIT1, c_frame, FALSE ); //現在のフレーム数を EditBox に書き込む
997                             frame_display(0);
998                             number = 0;//検出数を元に戻す.
999                             if(add_num == 0) MessageBox(NULL, "終了:自動更新はOFF でした。", "MessageBox", MB_OK);
1000
1001                            //最終的な閾値とフレーム数をフォルダ名に追加する.
1002                            wsprintf(detect_dir_c_frame, "%s_s%d_%d", detect_dir,threshold, c_frame);
1003                            rename(detect_dir,detect_dir_c_frame);
1004                            /*
1005                            if(rename(detect_dir,detect_dir_c_frame) == 0){
1006                                    MessageBox(NULL, "フォルダ名変更した。","debug", MB_OK);
1007                            }else
1008                                    MessageBox(NULL, "フォルダ名変更出来ない。","debug", MB_OK);
1009                            */
1010
1011                            return TRUE;
1012                    }
1013            }//m 自動更新終了.
1014            fclose(file);
1015            _fseeki64 ( fp, (__int64)headersize + (__int64)c_frame * (__int64)ImageSize * 2, SEEK_SET );
1016            ser_read(fp);
1017            //SetDlgItemInt( hDlg, IDC_EDITjikkou, repeat * hani, FALSE ); //実行フレーム数をEditBox に書き込む
1018            SetDlgItemInt( hDlg, IDC_EDIT1, c_frame, FALSE ); //現在のフレーム数を EditBox に書き込む
1019            frame_display(0);
1020
1021            if(add_num == 0) MessageBox(NULL, "終了", "MessageBox", MB_OK);
1022    return TRUE;
1023
1024    //case IDC_BUTTON5:   //フレームを保存
1025            //wsprintf(fname, "frame_camera_%1d_%08d.csv", camera_no, date);
1026            //fopen_s(&fp, fname, "w" );
1027            //for (k = 0; k < count ; k++) {
1028            // if ((frame[k] != frame[k-1])&&(frame[k] != 0)){
1029            //  fprintf (fp, "%6d\n", frame[k]);
1030            // }
1031            //}
1032            //fclose(fp);
1033            //MessageBox(NULL, "保存しました", "MessageBox", MB_OK);
1034            //return TRUE;
1035
1036    case IDCANCEL:    //右上の×クリックで終了
1037            PostQuitMessage(0);
1038            DestroyWindow (hDlg);
1039            //DestroyWindow (hImage1);
1040            DestroyWindow (g_img.hwin);
1041            fclose(fp);
1042            free ( value16 );
```

```
1043            free ( value8 );
1044            free ( value16_tmp);
1045            free (x2heikin);
1046            free (ave);
1047            free (sigma);
1048            return TRUE;
1049    }
1050    return TRUE;
1051    }
1052    return FALSE;
1053    }
1054
1055            //データウィンドウプロシージャ
1056            LRESULT CALLBACK WndProc0( HWND hImage0, UINT msg, WPARAM wp, LPARAM lp )
1057            {
1058            switch (msg) {
1059            case WM_KEYDOWN:
1060                    if(GetAsyncKeyState (VK_RIGHT))//→を押したら次フレーム表示
1061                            SendMessage (hDlg, WM_COMMAND, MAKEWPARAM(IDC_BUTTON_NEXT, 0), 0);
1062                    if(GetAsyncKeyState (VK_LEFT))//→を押したら次フレーム表示
1063                            SendMessage (hDlg, WM_COMMAND, MAKEWPARAM(IDC_BUTTON_PREV, 0), 0);
1064                    break;
1065            case WM_PAINT:        // ビットマップ全体を描画する
1066                    //MessageBox(NULL,"WM_PAINT","debug",MB_OK);
1067                    HDC hdc0, hdc_mem0;      //デバイスコンテキストおよびメモリデバイスコンテキストへのハンドル
1068                    PAINTSTRUCT ps0;
1069                    HBITMAP hBmp0;       //ビットマップへのハンドル
1070
1071                    hdc0 = BeginPaint( hImage0, &ps0 );  //デバイスコンテキストを取得する
1072                    hdc_mem0 = CreateCompatibleDC( hdc0 ); //メモリデバイスコンテキストを取得する
1073                    hBmp0 = CreateDIBitmap( hdc0, &(g_img.bmp_ih ), CBM_INIT, g_img.lpBmpData, //フレーム全体表
                                示のビットマップを作る
1074                                    (BITMAPINFO *)(&( g_img.bmp_ih )), DIB_RGB_COLORS );
1075                    SelectObject(hdc_mem0, hBmp0);   //メモリデバイスコンテキストにビットマップを入れる
1076
1077                    StretchBlt(hdc0, 0, 0, ImageWidth2, ImageHeight2, hdc_mem0, 0, 0, ImageWidth, ImageHeight,
                                SRCCOPY); //拡大して表
                                示
1078
1079                    DeleteDC( hdc0 );
1080                    DeleteDC( hdc_mem0 );
1081                    DeleteObject( hBmp0 );
1082                    EndPaint( hImage0, &ps0 );
1083
1084                    return 0;
1085
1086            case WM_MOUSEMOVE:
1087                    return 0;
1088            default:
1089                    return DefWindowProc( hImage0, msg, wp, lp );
1090            }
1091            return 0;
1092            }
```

function_file.cpp

```
1   #include "header.h''
2   //ファイルのパス名を得る関数
3   char* file_open()
4   {
5   OPENFILENAME fname;
6   static char  fn[256];
7   static char  filefilter[] = "ser ファイル(*.ser)\0*.ser\0"
8   "すべてのファイル (*.*)\0*.*\0\0";
9
10  memset( &fname, 0, sizeof(OPENFILENAME) );
11  fname.lStructSize = sizeof(OPENFILENAME);
12  fname.lpstrFilter = filefilter;
13  fname.nFilterIndex = 1;
14  fname.lpstrFile = fn;
15  fname.nMaxFile = sizeof(fn);
16  fname.Flags = OFN_FILEMUSTEXIST | OFN_HIDEREADONLY;
17
18  if( !GetOpenFileName( &fname ) ) return NULL;
19
20  return fn;
21  }
22
23  //ser ファイルのヘッダー(178 Byte)を読込み、画像の幅、高さ、1画素のビット数、全フレーム数を得る関数
24  FILE* read_header( FILE *fp, BOOL mkd, char* folder_name)//mkd は 1 の時、ディレクトリを作成する。
25  {
26  //Tech(folder_name);
27  char cbuf[100]={};
28  int ibuf;
29  unsigned long long ibuf64;
30  const int strsize = 1000;
31  char str[strsize]={};
```

```
32  char tmp[100]={};
33  fseek ( fp, 0, SEEK_SET );
34  fread ( &cbuf, 14, 1, fp ); //Header 中の FileID(14byte)を読み込む
35  wsprintf ( tmp, "FileID(14byte) : %s\x0d\x0a", cbuf );
36  strcat_s ( str, strsize, tmp );
37  fread ( &ibuf, 4, 1, fp );
38  wsprintf ( cbuf, "LuID(int) : %d\x0d\x0a", ibuf );
39  strcat_s ( str, strsize, cbuf );     // str に cbuf を追加する
40
41  fread ( &ibuf, 4, 1, fp );
42  wsprintf ( cbuf, "ColorID(int) : %d\x0d\x0a", ibuf );
43  strcat_s ( str, strsize, cbuf );     // str に cbuf を追加する
44
45  fread ( &ibuf, 4, 1, fp );
46  wsprintf ( cbuf, "LittleEndian(int) : %d\x0d\x0a", ibuf );
47  strcat_s ( str, strsize, cbuf );     // str に cbuf を追加する
48
49  fread ( &ibuf, 4, 1, fp );
50  ImageWidth = ibuf;
51  wsprintf ( cbuf, "ImageWidth(int) : %d\x0d\x0a", ibuf );
52  strcat_s ( str, strsize, cbuf );     // str に cbuf を追加する
53
54  fread ( &ibuf, 4, 1, fp );
55  ImageHeight = ibuf;
56  wsprintf ( cbuf, "ImageHeight(int) : %d\x0d\x0a", ibuf );      //高さ
57  strcat_s ( str, strsize, cbuf );     // str に cbuf を追加する
58
59  fread ( &ibuf, 4, 1, fp );
60  PixelDepthPerPlane = ibuf;
61  wsprintf ( cbuf, "PixelDepthPerPlane(int) : %d\x0d\x0a", ibuf );     //幅
62  strcat_s ( str, strsize, cbuf );     // str に cbuf を追加する
63
64  fread ( &ibuf, 4, 1, fp );
65  FrameCount = ibuf;
66  wsprintf ( cbuf, "FrameCount(int) : %d\x0d\x0a", ibuf );      //フレーム数
67  strcat_s ( str, strsize, cbuf );     // str に cbuf を追加する
68
69  fread ( &cbuf, 40, 1, fp );
70  wsprintf ( tmp, "Observer(40byte) : %s\x0d\x0a", cbuf );
71  strcat_s ( str, strsize, tmp );     // str に cbuf を追加する
72
73  fread ( &cbuf, 40, 1, fp );
74  wsprintf ( tmp, "Instrument(40byte) : %s\x0d\x0a", cbuf );
75  strcat_s ( str, strsize, tmp );     // str に cbuf を追加する
76
77  fread ( &cbuf, 40, 1, fp );
78  wsprintf ( tmp, "Telescope(40byte) : %s\x0d\x0a", cbuf );
79  strcat_s ( str, strsize, tmp );     // str に cbuf を追加する
80
81  fread ( &ibuf64, 8, 1, fp );
82  wsprintf ( cbuf, "Datetime:%I64x\x0d\x0a", ibuf64 );
83  strcat_s ( str, strsize, cbuf );
84
85  fread ( &ibuf64, 8, 1, fp );
86  wsprintf ( cbuf, "Datetime␣UTC:%I64x\x0d\x0a", ibuf64 );
87  strcat_s ( str, strsize, cbuf );
88
89  if(mkd){
90  //char foldername[100];
91  //wsprintf(foldername, "Detection_%s", &fname);
92  //_mkdir(foldername);
93  char detect_dir[100]={};
94  char detect_result_path[100]={};
95  wsprintf(detect_dir, "検出_%s", folder_name);
96  //Tech(detect_dir);
97  _mkdir(detect_dir);
98  FILE *file;
99  wsprintf(detect_result_path, "%s\\SER_Detection_Result.txt", detect_dir);
100 fopen_s(&file,detect_result_path,"w");
101 fprintf(file, "SER␣File␣Header␣Information----------------------------\n");
102 fprintf(file, str);
103 fprintf(file, "------------------------------------------------------\n\n");
104
105 fclose(file);
106 }//else    MessageBox(NULL, str, "SER File Details", MB_OK);
107 return fp;
108 }
109
110 void ser_read(FILE *fp){ //Value16 に画素データを格納
111 if( PixelDepthPerPlane == 16 ){
112 fread(value16, sizeof(uint16), ImageSize,fp);//1枚分のデータを読み込む
113 for(int i =0; i < ImageHeight; i++){
114 for(int j = 0; j< ImageWidth; j++){
115 value16[ImageWidth*i +j] = value16[ImageWidth*i +j] >>4;}//ビットシフト
116 }
117 if(Button_GetCheck(hRadio_median_on) == BST_CHECKED){//
118 for(int i =0; i < ImageHeight; i++){
119 for(int j = 0; j< ImageWidth; j++){
```

```
value16_tmp[ImageWidth*i +j] = value16[ImageWidth*i +j];
}
}
ser_median();
}
}
}
void ser_median(){
uint16 sort[9]={};
uint16 temp;
int i, j, k, l,m;
for( i = 1; i < ImageHeight - 1; i++){//画面端はフィルター適応されない。
for( j = 1; j< ImageWidth - 1; j++){
for(k = 0; k < 3; k++){
sort[k] = value16_tmp[ImageWidth*(i-1) +j-1 + k];//上段 3つ
sort[k+3] = value16_tmp[ImageWidth*i +j-1 + k];//中段 3つ
sort[k+6] = value16_tmp[ImageWidth*(i+1) +j-1 + k];//下段 3つ
}
//3*3の画素を挿入ソート
for (l = 1; l < 9; l ++) { /* l 番目の要素をソート済みの配列に挿入 */
temp = sort[l]; /* l 番目の要素を temp に保存 */
for (m = l; m > 0 && sort[m-1] > temp; m--) /* このループで */
sort[m] = sort[m-1]; /* temp を挿入する位置を決める */
sort[m] = temp; /* temp を挿入 */
}
value16[ImageWidth*i+j] = sort[4];//中央値を代入する.
}
}
return;
}
//保存用ファイル選択
char* file_save( int file_type )
{
OPENFILENAME fname;
static char  fn[256];
static char  *filefilter;

if ( file_type == 1 ) filefilter = "bmp ファイル(*.bmp)\0*.bmp\0"
"すべてのファイル (*.*)\0*.*\0\0";
if ( file_type == 2 ) filefilter = "すべてのファイル (*.*)\0*.*\0\0";

memset( &fname, 0, sizeof(OPENFILENAME) );
fname.lStructSize = sizeof(OPENFILENAME);
fname.lpstrFilter = filefilter;
fname.nFilterIndex = 1;
fname.lpstrFile = fn;
fname.nMaxFile = sizeof(fn);
fname.Flags = OFN_FILEMUSTEXIST | OFN_HIDEREADONLY;

if( !GetSaveFileName( &fname ) ) return NULL;

return fn;
}

void Tech(char * ff){
MessageBox(NULL, ff, "debug",MB_OK);//フレームを表示する
}
```

function_image.cpp

```
1
2   #include "header.h"
3
4   //表示用ビットマップを準備する (大きさは読み込んだTIFF ファイルの縦横半分)
5   void process0()
6   {
7   free(g_img.lpBmpData);
8   g_img.bmp_ih.biSize = sizeof(BITMAPINFOHEADER);
9   g_img.bmp_ih.biWidth = ImageWidth;
10  g_img.bmp_ih.biHeight = -ImageHeight;  //マイナスにして左上隅を原点とする
11  g_img.bmp_ih.biPlanes = 1;
12  g_img.bmp_ih.biBitCount = 24;
13  g_img.bmp_ih.biCompression = BI_RGB;
14  g_img.bmp_ih.biSizeImage = ImageWidth * ImageHeight * 4; //Width*Height*BytesPerPixel
15  g_img.bmp_ih.biXPelsPerMeter = 3704;
16  g_img.bmp_ih.biYPelsPerMeter = 3704;
17  g_img.bmp_ih.biClrUsed = 0;
18  g_img.bmp_ih.biClrImportant = 0;
19
20  g_img.lpBmpData = (BYTE *)malloc( g_img.bmp_ih.biSizeImage);
21
22  return;
23  }
24
25  //全体画面を表示する関数
26  void frame_display( int flag_disp )
```

```
{
BOOL check;   //GetDlgItemInt のチェックに使う
double bright;  //表示する明るさの実数値
int i, j, ij;  //色々に使う
int offset =0;
int mag_brt = GetDlgItemInt( hDlg, IDC_EDIT_MAG_BRT, &check, TRUE ); //明るさ倍率の 1/1000を得る
if( !check ) return;
for (i=0; i<ImageHeight; i=i+1) {
for (j=0; j<ImageWidth; j=j+1) {
ij = i*ImageWidth + j;    //ij は生画像の最初から何バイト目かを示す
bright = (double)value16[ij] * (double)mag_brt/1000. + (double)offset; //生画像表示
if ( bright < 0. ) bright = 0.;
if ( bright > 255. ) bright = 255.;

g_img.lpBmpData[ij*3+0] = (byte)bright; //ビットマップの B
g_img.lpBmpData[ij*3+1] = (byte)bright; //ビットマップの G
g_img.lpBmpData[ij*3+2] = (byte)bright; //ビットマップの R
}
}

InvalidateRect( g_img.hwin, NULL, FALSE); //フレームを表示する

return;
}
```

header.h

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <shlobj.h>//フォルダを開く時に使用
#include <direct.h>
#include <Windows.h>
#include <Windowsx.h>
#include <math.h>
#include <Shlwapi.h>//パスを弄る時に使用
#pragma comment(lib, "shlwapi.lib")

//////////////////////////////////////////////////リストビュー
#include <windows.h>
#include <commctrl.h>
#include "resource.h"
#pragma comment(lib, "comctl32.lib")

////////////////////////////////////////////////
#include "resource.h"
#include "C:\tiff-4.0.6\libtiff\tiff.h"  //TIFF 読込に必要
#include "C:\tiff-4.0.6\libtiff\tiffio.h" //TIFF 読込に必要

typedef struct{      //IMG0 という構造体を定義する
HINSTANCE hi0;
HWND hwin;   //自分のウィンドウハンドル
BYTE *lpBmpData; //ビットマップ画像のデータ部分
BITMAPINFOHEADER bmp_ih;  //ビットマップインフォヘッダー
char   *fname;
int    fsize;
} IMG0;

#define LIST_ITEM_MAX 40//開けるファイルの上限数
typedef struct{
char list_fname[30];//ファイル名
char file_path[LIST_ITEM_MAX][MAX_PATH];//パス名
char ch_start_no[10];//開始フレーム
char ch_end_no[10];//終了フレーム
char ch_limit_no[10];
char ch_threshold[10];
int start_no[LIST_ITEM_MAX];
int end_no[LIST_ITEM_MAX];
int limit_no[LIST_ITEM_MAX];
int threshold[LIST_ITEM_MAX];
int allframe[LIST_ITEM_MAX];
} DLIST;

#define SQUARE_LIMIT 50
ATOM InitApp0( HINSTANCE );   //全体画面用ウィンドウクラスを登録する関数
ATOM InitApp1( HINSTANCE );   //拡大画面用ウィンドウクラスを登録する関数
BOOL InitInstance0(HINSTANCE, int); //全体画面用ウィンドウを作成する関数
BOOL InitInstance1(HINSTANCE, int); //拡大画面用ウィンドウを作成する関数

BOOL CALLBACK dlgproc( HWND, UINT, WPARAM, LPARAM ); //ダイアログウィンドウのコールバック関数
LRESULT CALLBACK WndProc0( HWND, UINT, WPARAM, LPARAM ); //全体画面ウィンドウのコールバック関数
LRESULT CALLBACK WndProc1( HWND, UINT, WPARAM, LPARAM ); //拡大画面ウィンドウのコールバック関数
char* file_save( HWND );     //全体画像やカウント値を保存するファイルを開く関数
//void frame_display( HWND, IMG0, int );  //フレームを全体画面に表示する関数
void DrawRect(HWND, POINTS, POINTS, int); //拡大画面にカウント領域の四角を描く関数
char* file_open();     //ファイルのパス名を得る関数
```

```
60 //FILE* read_header( FILE* , BOOL, char* );  //ser ファイルのヘッダー(178 Byte)を読込み、画像の幅、高さ、1画素のビット数、
       全フレーム数を得る関数
61 FILE* read_header( FILE* , BOOL, char*);  //ser ファイルのヘッダー(178 Byte)を読込み、画像の幅、高さ、1画素のビット数、
       全フレーム数を得る関数
62 void process0();    //表示用ビットマップを準備する (大きさは読み込んだTIFF ファイルの縦横半分)
63 void frame_display( int );    //フレームを全体画面に表示する関数
64 void ser_read(FILE*);
65 void ser_median();
66 void Tech(char*);
67 int Count_Save ( int c_save, int c_f[300], int xy_for_count[3][4][300], uint64 count[3][300] );
68 extern HWND hDlg, hImage1;
69 extern HWND hRadio_median_on, hRadio_median_off;
70 extern uint16 *value16;    //16it_ser 画素値の画像全体分の配列
71 extern uint16 *value16_tmp;    //16it_ser 画素値の画像全体分の配列
72 extern uint8 *value8;    // 8it_ser 画素値の画像全体分の配列
73 extern double *x2heikin; //平均計算のために輝度値の合計を入れる
74 extern double *ave;
75 extern double *sigma; //平均値を入れる
76 extern int ImageWidth, ImageHeight, PixelDepthPerPlane, FrameCount;
77 extern int ImageWidth2, ImageHeight2, ImageSize; //画像の幅と高さの 1/2、および 1フレームの全画素数
78 extern IMG0 g_img;
79 extern char detection_path[100];
```

resource.h

```
1  #define IDR_MENU1 102
2  #define IDC_BUTTON1 1001
3  #define IDC_BUTTON_FRAME 1002
4  #define IDC_BUTTON_PREV 1003
5  #define IDC_BUTTON_NEXT 1004
6  #define IDC_HIST 1005
7  #define IDC_EDIT1 1006
8  #define IDC_BUTTON_FOLDER_OPEN 1007
9  #define IDC_EDIThani 1009
10 #define IDC_EDITstart 1012
11 #define IDC_EDITallframe 1013
12 #define IDC_EDITx 1015
13 #define IDC_EDITy 1016
14 #define IDC_EDITblue 1017
15 #define IDC_EDITgreen 1018
16 #define IDC_RADIO1 1019
17 #define IDC_RADIO2 1020
18 #define IDC_RADIO3 1021
19 #define IDC_EDITred 1022
20 #define IDC_EDITthreshold 1023
21 #define IDC_EDITjikkou 1025
22 #define IDC_EDITend 1026
23 #define IDC_RADIO_MEDIAN_ON 1027
24 #define IDC_RADIO_MEDIAN_OFF 1028
25 #define IDC_RADIO_AUTO_UPDATE_ON 1029
26 #define IDC_RADIO_AUTO_UPDATE_OFF 1030
27 #define IDC_EDIT 1031
28 #define IDC_EDITue 1062
29 #define IDC_EDIThidari 1063
30 #define IDC_EDITmigi 1064
31 #define IDC_EDITshita 1065
32 #define IDC_EDITtotal 1066
33 #define IDC_EDIT_MAG_BRT 1066
34 #define IDC_CHECK1 1067
35 #define IDC_CHECK2 1068
36 #define IDC_BUTTON5 1069
37 #define IDC_EDIT_DETECT_LIMIT 1070
38 #define IDC_LIST1 1072
39 #define IDC_LIST2 1076
40 #define IDC_BUTTON6 1077
41 #define IDC_BUTTON_LISTHIST 1077
42 #define IDC_BUTTON_ADD 1078
43 #define IDC_BUTTON_REMOVE 1079
44 #define IDC_EDIT2 1081
45 #define IDC_EDIT_PATH 1081
46 #define IDC_BUTTON_ITEM_EDIT 1082
47 #define IDC_SPIN1 1083
48
49 // Next default values for new objects
50 //
51 #ifdef APSTUDIO_INVOKED
52 #ifndef APSTUDIO_READONLY_SYMBOLS
53 #define _APS_NEXT_RESOURCE_VALUE 104
54 #define _APS_NEXT_COMMAND_VALUE 40001
55 #define _APS_NEXT_CONTROL_VALUE 1084
56 #define _APS_NEXT_SYMED_VALUE 101
57 #endif
58 #endif
```

newdetect.rc

```
1  // Microsoft Visual C++ generated resource script.
2  //
3  #include "resource.h"
4
5  #define APSTUDIO_READONLY_SYMBOLS
6  ////////////////////////////////////////////////////////////////////////////
7  //
8  // Generated from the TEXTINCLUDE 2 resource.
9  //
10 #include "afxres.h"
11
12 ////////////////////////////////////////////////////////////////////////////
13 #undef APSTUDIO_READONLY_SYMBOLS
14
15 ////////////////////////////////////////////////////////////////////////////
16 // 日本語 (日本) resources
17
18 #if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_JPN)
19 LANGUAGE LANG_JAPANESE, SUBLANG_DEFAULT
20 #pragma code_page(932)
21
22 #ifdef APSTUDIO_INVOKED
23 ////////////////////////////////////////////////////////////////////////////
24 //
25 // TEXTINCLUDE
26 //
27
28 1 TEXTINCLUDE
29 BEGIN
30 "resource.h\0"
31 END
32
33 2 TEXTINCLUDE
34 BEGIN
35 "#include␣""afxres.h""\r\n"
36 "\0"
37 END
38
39 3 TEXTINCLUDE
40 BEGIN
41 "\r\n"
42 "\0"
43 END
44
45 #endif // APSTUDIO_INVOKED
46
47 ////////////////////////////////////////////////////////////////////////////
48 //
49 // Dialog
50 //
51 IDD_DIALOG1 DIALOGEX 754, 0, 267, 395
52 STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_VISIBLE | WS_CAPTION |
       WS_SYSMENU
53 CAPTION "Serdetect.exe␣Ver.1.17"
54 FONT 8, "MS␣Shell␣Dlg", 400, 0, 0x1
55 BEGIN
56 PUSHBUTTON "指定された\n フレームを表示",IDC_BUTTON_FRAME,68,36,56,24,BS_MULTILINE
57 PUSHBUTTON "戻る",IDC_BUTTON_PREV,8,65,50,15
58 PUSHBUTTON "進む",IDC_BUTTON_NEXT,68,65,50,15
59 EDITTEXT IDC_EDIT1,8,47,48,12,ES_AUTOHSCROLL
60 CTEXT "フレーム番号",IDC_STATIC,10,38,43,8
61 EDITTEXT IDC_EDIThani,90,183,24,14,ES_AUTOHSCROLL
62 LTEXT "処理間隔",IDC_STATIC,41,185,43,8,0,WS_EX_RIGHT
63 LTEXT "上下左右の検出条件",IDC_STATIC,159,126,74,8
64 EDITTEXT IDC_EDITthreshold,90,203,24,13,ES_AUTOHSCROLL
65 LTEXT "/10␣σ␣",IDC_STATIC,120,204,36,13
66 EDITTEXT IDC_EDITend,90,141,36,14,ES_AUTOHSCROLL
67 LTEXT "終了フレーム番号",IDC_STATIC,22,146,62,9,0,WS_EX_RIGHT
68 LTEXT "開始フレーム番号",IDC_STATIC,22,126,62,9,0,WS_EX_RIGHT
69 GROUPBOX "条件を設定",IDC_STATIC,8,114,252,108
70 EDITTEXT IDC_EDITue,188,150,17,14,ES_AUTOHSCROLL
71 EDITTEXT IDC_EDIThidari,160,167,17,14,ES_AUTOHSCROLL
72 EDITTEXT IDC_EDITmigi,214,167,17,14,ES_AUTOHSCROLL
73 EDITTEXT IDC_EDITshita,188,185,17,14,ES_AUTOHSCROLL
74 LTEXT "上",IDC_STATIC,192,138,9,8
75 LTEXT "下\t\t\t\tif␣(detect␣==␣1){␣",IDC_STATIC,193,202,8,8
76 LTEXT "右",IDC_STATIC,206,170,8,8
77 LTEXT "左",IDC_STATIC,150,170,9,8
78 EDITTEXT IDC_EDIT_MAG_BRT,214,12,30,14,ES_AUTOHSCROLL
79 LTEXT "全体画面の明るさ",IDC_STATIC,150,15,59,8
80 CONTROL "メディアンフィルタ␣ON",IDC_RADIO_MEDIAN_ON,"Button",BS_AUTORADIOBUTTON | BS_LEFTTEXT
       ,150,42,84,10,WS_EX_RIGHT
81 CONTROL "メディアンフィルタ␣OFF",IDC_RADIO_MEDIAN_OFF,"Button",BS_AUTORADIOBUTTON |
       BS_LEFTTEXT,152,55,82,10,WS_EX_RIGHT
82 GROUPBOX "メディアンフィルタの有無",IDC_STATIC,144,30,102,41
83 EDITTEXT IDC_EDIT_DETECT_LIMIT,90,161,24,14,ES_AUTOHSCROLL
84 LTEXT "検出上限フレーム数",IDC_STATIC,12,166,72,9,0,WS_EX_RIGHT
```

```
 85  LTEXT "総フレーム数",IDC_STATIC,18,93,42,8
 86  EDITTEXT IDC_EDITallframe,69,90,49,14,ES_AUTOHSCROLL
 87  CONTROL "自動更新␣ON",IDC_RADIO_AUTO_UPDATE_ON,"Button",BS_AUTORADIOBUTTON | BS_LEFTTEXT
         ,150,84,84,10,WS_EX_RIGHT
 88  CONTROL "自動更新␣OFF",IDC_RADIO_AUTO_UPDATE_OFF,"Button",BS_AUTORADIOBUTTON |
         BS_LEFTTEXT,171,97,63,10,WS_EX_RIGHT
 89  GROUPBOX "検出条件を自動で更新",IDC_STATIC,144,72,102,41
 90  LISTBOX IDC_LIST1,0,225,88,151,LBS_SORT | LBS_NOINTEGRALHEIGHT | LBS_MULTICOLUMN |
         WS_VSCROLL | WS_HSCROLL | WS_TABSTOP,WS_EX_STATICEDGE
 91  PUSHBUTTON "フォルダを開く",IDC_BUTTON_FOLDER_OPEN,8,6,56,24
 92  CONTROL "",IDC_LIST2,"SysListView32",LVS_REPORT | LVS_SHOWSELALWAYS | LVS_ALIGNLEFT |
         WS_BORDER | WS_TABSTOP,116,225,149,151,WS_EX_STATICEDGE
 93  PUSHBUTTON "検出",IDC_BUTTON_LISTHIST,90,315,24,16
 94  PUSHBUTTON "追加",IDC_BUTTON_ADD,90,252,24,16
 95  PUSHBUTTON "削除",IDC_BUTTON_REMOVE,90,273,24,16
 96  EDITTEXT IDC_EDIT_PATH,0,380,266,15,ES_AUTOHSCROLL
 97  PUSHBUTTON "編集",IDC_BUTTON_ITEM_EDIT,90,294,24,16
 98  EDITTEXT IDC_EDITstart,90,121,36,14,ES_AUTOHSCROLL
 99  LTEXT "閾値",IDC_STATIC,54,204,30,8,0,WS_EX_RIGHT
100  LTEXT "%",IDC_STATIC,178,170,8,8
101  LTEXT "%",IDC_STATIC,206,188,8,8
102  LTEXT "%",IDC_STATIC,208,153,8,8
103  LTEXT "%",IDC_STATIC,232,170,8,8
104  CONTROL "",IDC_SPIN1,"msctls_updown32",UDS_ARROWKEYS,244,12,11,14
105  END
106  /////////////////////////////////////////////////////////////////////////////
107  //
108  // DESIGNINFO
109  //
110  #ifdef APSTUDIO_INVOKED
111  GUIDELINES DESIGNINFO
112  BEGIN
113  "IDD_DIALOG1", DIALOG
114  BEGIN
115  BOTTOMMARGIN, 355
116  END
117  END
118  #endif // APSTUDIO_INVOKED
119
120  #endif // 日本語 (日本) resources
121  /////////////////////////////////////////////////////////////////////////////
122  #ifndef APSTUDIO_INVOKED
123  /////////////////////////////////////////////////////////////////////////////
124  //
125  // Generated from the TEXTINCLUDE 3 resource.
126  //
127  /////////////////////////////////////////////////////////////////////////////
128  #endif // not APSTUDIO_INVOKED
```